computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

MULTI-AGENT PARALLEL IMPLEMENTATION OF PHOTOMASK SIMULATION IN PHOTOLITHOGRAPHY

Syarhei M. Avakaw [1)], Alexander A. Doudkin [2)], Alexander V. Inyutin [2)], Aleksey V. Otwagin [2,3)], Vladislav A. Rusetsky [1)]

[1)] Research and Production Republican Unitary Enterprise "KBTEM-OMO" of Planar Corporation, 2 Partizansky Ave, Minsk, Belarus, mve@kbtem.avilink.net
[2)] United Institute of Informatics Problems of NAS of Belarus, Minsk, doudkin@newman.bas-net.by
[3)] Belarusian State University of Informatics and Radioelectronics, 6 P. Brovka st, Minsk, Belarus, otwagin@bsuir.by

Abstract: *A framework for paralleling aerial image simulation in photolithography is proposed. Initial data for the simulation representing photomask are considered as a data stream that is processed by a multi-agent computing system. A parallel image processing is based on a graph model of a parallel algorithm. The algorithm is constructed from individual computing operations in a special visual editor. Then the visual representation is converted into XML, which is interpreted by the multi-agent system based on MPI. The system performs run-time dynamic optimization of calculations using an algorithm of virtual associative network. The proposed framework gives a possibility to design and analyze parallel algorithms and to adapt them to architecture of the computing cluster.*

Keywords: *Aerial Image Simulation, Multi-agent, Integrated Circuit, Photolithography, Parallel Algorithm.*

1. INTRODUCTION

A photolithography process is a major step in conversion of integrated circuit (IC) layout pattern on a surface of a semiconductor wafer. A simulation of the lithography process is very complicated. It can be roughly simulated in two steps:

1) mask shapes are projected into a photoresist as an aerial image;

2) a distribution of an absorbed intensity of a emission in the photoresist are calculated and patterned based on the aerial image intensity.

The image projection is simulated by the Hopkins equation [1], which is a four-dimensional integral. This calculation is too slow to simulate across the full chip and parallel algorithms can be used to speed up the simulation [2,3]. Currently, the known number of photolithography simulation software systems is implemented both on clusters of multiprocessor personal computers and workstations [4-6, 7] and supercomputers [9]. Hardware-accelerated computational lithography tools are also built [8]. The need of solving additional tasks for planning and optimizing the structure of a parallel application in the design process prevents their widespread use.

In many cases, a development of the parallel applications is carried out based on existing sequential algorithms and their composition. Computational operations as parts of the parallel algorithm often have a universal character and can be applied to various problems of information processing. Implementations of the operations in the portable forms allow going to component design, when the program is constructed from large blocks.

In addition, a process of a parallel program execution requires modern methods of planning and optimization of load characteristics of computational nodes. In many cases, the nodes of parallel systems have heterogeneous characteristics, both spatial and temporal. For an effective implementation of parallel programs such systems should provide tools for organizing and monitoring parallel computing and its dynamic reconfiguration.

There are a number of machine vision systems that use parallel and distributed processing [10-13]. Different technologies such as CORBA [13] or a multi-agent approach [10] are used as a architectural core of these systems.

We propose to use for designing and organizing parallel computations an integrated set of tools (framework) that includes a visual editor, a compiler, an optimization system and a parallel computing support system based on MPI [14]. Having framework for design, analysis and planning

of parallel applications and built-in specific mechanisms for the implementation of parallelism, IC designer can significantly accelerate the processing flow of photomask and layout images in the operational analysis of IC. MPI makes our system is widely applicable to various parallel computers. The main features of the proposed framework are the following:

1) visual representation of a parallel application based on graph model of a computation. A concept of computational grains is used. The grains are independent modules developed in different programming languages (C + +, Java, MPI) and have a specific interface for integration into a parallel algorithm;

2) a support of a portability, implemented as libraries. The computing grains are added to the system algorithms at a run time;

3) static and dynamic optimization of the parallel applications using an algorithm of virtual associative network (VAN). This algorithm is a kind of hybrid genetic algorithm (GA) and provides a quick search for a solution close to optimal. It has two modifications: the first one is used in the analysis phase of the design (the static optimization), the second one – on the stage of program execution (the dynamic optimization);

4) assignment of operations of the parallel application on the computational nodes (processors) of a computer system, taking into account information obtained during the optimization.

The paper describes the implementation of parallel simulating of image formation in photoresist wafer during photolithography. Init data for processing are the images of the original topology of VLSI photomasks. Results of simulating give a possibility to do a subsequent automatic mask inspection and determine a significance of photolithography defects. The defects of the topology are significant if they lead to the formation of defects on the wafer that should be corrected at the stage of the production.

The task of simulation of an aerial image is calculation the light intensity distribution of the wafer surface and to obtain a latent image with regard to characteristics of the optical system and lighting conditions. Section 2 represents an algorithm that simulates the aerial image on the photoresist. In Section 3 we consider the problem of parallel processing and describe a graph model representation of the parallel algorithm based on concept of computational grains. Section 4 describes the basic elements of the computational platform and their interaction in the development of parallel applications. Section 5 represents an example of an implementation of the simulation algorithm and some experimental results of the optimization process.

2. ALGORITHM FOR SIMULATING AERIAL IMAGE ON THE PHOTORESIST

Algorithm for simulating the image on the photoresist surface is composed of the following steps [15, 16]:

- calculation of a pupillary function;
- calculation of a vector amplitude of the object;
- calculation of a transfer matrix of the projection lens
- calculation of two-dimensional distribution of intensity in a given position of the plane
- calculation of two-dimensional distribution of intensity in different positions of the plane;
- calculation of image intensity in semi-coherent light;
- calculation of the volume distribution of intensity.

The influence of the vector properties of light is taken into account by the so-called vector factors (multipliers) applied to the pupil function. Using the factors allowed describing the influence of the vector nature of electromagnetic waves on the image of thin periodic structures, whose dimensions are within the resolution of optical systems, significantly reducing the computation time of the aerial image.

To describe the effect of an anterior aperture of the optical system it is necessary to use the factor that accounts the diffraction of a plane linearly polarized wave at the input of the optical system. In this case, we consider the effect of the optical system to redistribute the energy in the spectrum regardless of the direction of polarization and the direction of wave propagation. Another factor takes into account the influence of the entrance aperture at the entrance of the optical system. Based on these data, we can simulate the effect of an influence of the numerical entrance aperture to the distribution of any Fourier component exactly as a vector field (a vector nature of electromagnetic waves is considered at the inlet and outlet of the optical system). As a result, it is possible to calculate the vector field of the image both in coherent and partially coherent light.

Using the factors allows describing the influence of a linearly polarized wave to the image quality. For the case non-polarized or partially polarized light it is necessary to use both electric and magnetic vectors which are implemented in the proposed algorithm.

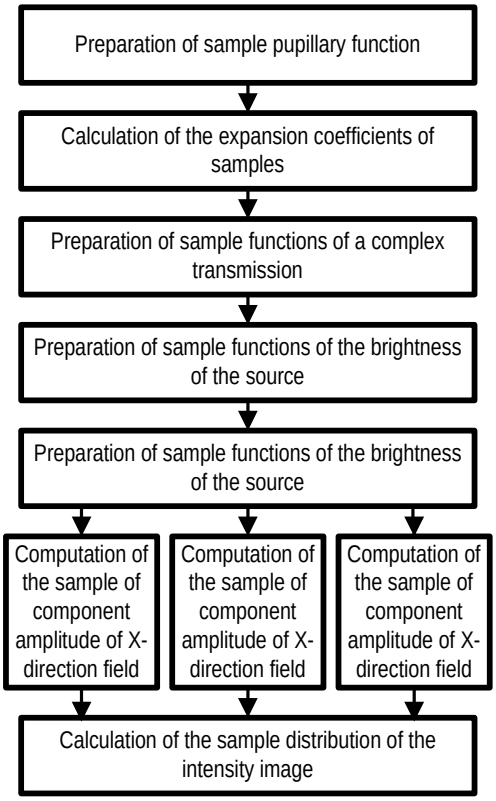

Fig. 1 –An algorithm for constructing the aerial image

The main features of the algorithm are the following.

1) Integrating the vector nature of the light field based on the wording of the electric and the magnetic vector of the amplitudes as functions of the three space Cartesian coordinates, as well as two coordinates in the pupil of the optical system. This formulation provides a correct account of the aberrations of the optical system and an influence of high numerical aperture to an image formation without a significant complication of the mathematical apparatus. In contrast to the currently used models, the proposed model is based on a strict conformity physical nature of the processes and much simpler, it is favorable for constructing fast algorithms.

2) Using the partial coherence theory the image intensity is calculated the most economical way based on a system of Eigen functions. Eigen functions are simulated by Zernike polynomials and are used to describe the mutual intensity of different pixels of the image. Such technique significantly reduces the number of integrals over the light source, a calculation of which remains to be the most time-consuming step in the simulation after applying the features mentioned above.

These principles supplement each other in developing the most efficient algorithm for calculating the intensity distribution of the aerial image and do not individually represent a value what they find together. The implementation of the algorithm in the form of single-process applications has shown the following: if the data level is large, then the processing time is unacceptably large (over 1 min / frame). Since all of the layout images are processed by a common program and may be called by a defect detection system at the same time, it is expedient to use a parallel computer system for the simultaneous processing of input data stream.

3. A PARALLEL APPLICATION MODEL AND A COMPUTATIONAL GRAIN CONCEPT

The basic principles of creation a graph-oriented parallel program representation are defined in [17]. The scenarios for data processing are represented 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 nodes, that represents decomposition of a parallel dataflow processing program on the separated operations $v_i \in V, 1 \le i \le N$;

E is a set of graph edges, that represents a precedence relation between operations in the scenario and determines a data transfer between these nodes, $\{e_{i,j} = (v_i, v_j)\} \in E, i = \overline{1,N}, j = \overline{1,N}, i \ne j$;

W is an operation cost matrix;

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 the scenarios, particular edges have an equal cost.

The development of a dataflow processing application includes the following three stages:

1) creation of a part of DAG scenario that describes logical structure of application;

2) assignment and editing of operations parameters of DAG scenario for each data type;

3) mapping of DAG scenario to cluster architecture.

Each computational operation in DAG scenario is realized as a separate unit called a grain. The grain uses specific interface for integration into framework and data exchange. A design of the grain makes possible a rapid adaptation of existing processing algorithms into a parallel application. These algorithms are transformed to objects that are capable to form their own calling context on the base of received parameters. Each operation interprets its parameter string by convenient way and converts the parameters to a variable name or to a constant value. The order and rules of a parameter transform are determined by an operation specification.

All operations work with a specific data storage mechanism that is incorporated into framework architecture. The storage realizes a shared memory abstraction for source data and results of processing. A variant of shared memory is realized on a shared file system that is common for many cluster architectures. A storage interface provides operations for writing and reading of data. There exists also an intermediate storage mechanism in local memory of each processor, where the results of this processor operation are stored. This one allows reducing time expenses for variable reading in case of repeated access.

The parameters of operation are read from storage. Each parameter is identified by its object name, represented as a string. Each parameter value is placed into corresponding internal grain variable, thus all parameters form a calling context. Further, the operation is executed and results of processing are placed in the storage. At this moment these values are accessible for other grains in parallel application.

The grains are collected in specific libraries that are dynamically linked into the parallel application. The grain is loaded from the library in due time and identified by the operation name. The realization of specific grain libraries from different classes of processing algorithms allows expanding the application area of the proposed framework.

An example of the parallel program graph is presented in Fig.2, where each operation is denoted as *Ci* with a cost vector. A cost of an information transfer between contiguous operations is equal for all data types. Some operations are strictly oriented on a specific processor while others can be placed on each processor in cluster. If the operation cost for some type of data is equal to zero, then this operation must be skipped for the selected type of data.

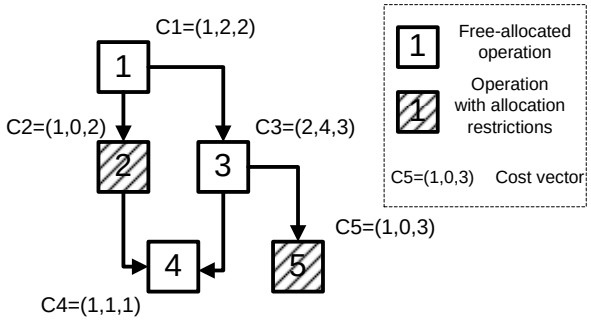

Fig. 2 – An example of program graph and denotation semantic

A matrix of restrictions $Z(O,P)$ is formed according to the following rules:

$Z(O,P) = 1$, if processor p allows execution of operation O;

$Z(O,P) = 0$, otherwise.

The matrix of restrictions is used in optimization procedures and prevents an erroneous allocation of the specified operations on some processors. The restrictions arise because of a heterogeneous cluster structure and different operations requirements.

A main task of a multi-agent system is a planning and an optimization of a parallel program execution with a simultaneous provision of reliable computations and guaranteed processing. There exist many algorithms of DAG scheduling that use various optimization techniques and heuristics. The techniques include priority based list scheduling, for example, the algorithms HLF (Highest Level First), LP (Longest Path) and CP (Critical Path) [18-20]. Another technique is a clusterization, and such algorithm, as DSC (Dominating Sequence Clustering) [21, 22] belongs to this technique. However all static scheduling algorithms are constructed mostly for special graph topologies, or use special constraints, such as a zero communication time between nodes or an unbounded number of processors. Because of a stochastic nature of input information the static scheduling approach can not realize an effective optimization for many cases of parallel processing.

Another perspective search techniques use an evolutionary optimization. These techniques are based on such algorithms, as tabu search [23], simulated annealing and genetic algorithms. GA combined with VAN algorithm is the most powerful technique among them. VAN algorithm is based on a concept of associations between the particular operations and dedicated processors [24]. Each operation O and processor P are associated by means of virtual link of strength $\omega_{O,P}$. Some structure of an associative memory, which consists of the associations, is constructed for optimization. This memory is learned by an experience, accumulated in a solution search process. VAN algorithm is based on GA representation of solutions in a form of population of chromosomes. Each chromosome represents a variant of program graph decomposition.

4. THE FRAMEWORK ARCHITECTURE AND AGENT BEHAVIOR

The architecture of agent framework for parallel processing is presented in Fig. 3.

The framework architecture is based on MPI and allows a fast communication between agents by means of an internal MPI virtual machine. A basic multi-agent structure is presented in Fig. 4.

An input for the multi-agent system is a parallel program graph and a set of data objects that are different by their types. The parallel graph is represented as XML file, which allows specifying all

the characteristics of separate operations for all types of data objects.

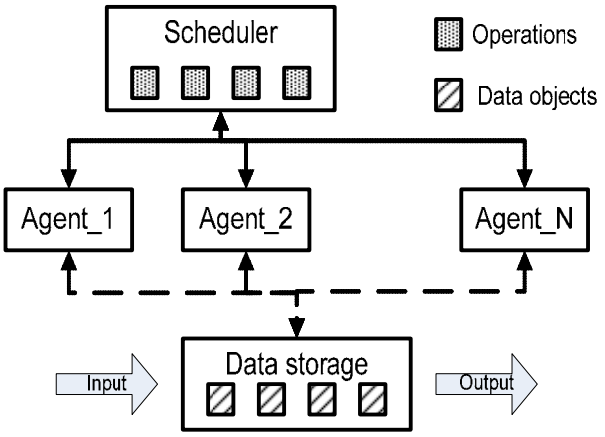

Fig. 3 – The architecture of parallel processing framework

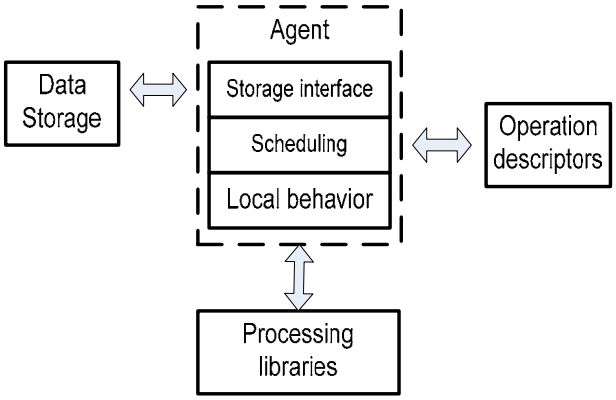

Fig. 4 – The internal structure of agent

Each agent performs parsing of the whole graph and builds internal structures that are used for an execution of specified operations with correct parameters for each object type. These operations are represented by descriptors, which are used by a scheduler to control precedence relations and an overall process.

The data object is represented by a descriptor too. The descriptor contains an identifier, type attributes and some additional information, for example, a name of data file, which contains information for this object. When an operation requires additional data for processing, this descriptor must be extended for specified applications 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 for an interaction with MPI facilities for messaging. The code is included in a message transfer interface that is extensible and allows the use of alternative message transport systems.

Data objects are stored in a shared data storage that is realized as descriptor storage. Each descriptor is linked with a universal container for storing of different data objects. The storage interface allows interaction with global storage for each agent in the system. This interface has some facilities for an object search, on-demand loading of remote objects and deletion of unused objects from the storage. Each agent has a local copy of the storage and uses it as a write-through cache.

The purpose of the scheduler interface is a processing and a scheduling control. It contains a special component, which is called a scheduler and makes decisions about the next processing operation that must be placed in a descriptor queue. All descriptors of the operations for processed objects are stored in the descriptor storage that contains three sets of descriptors: ready, working and finished pools. The scheduler chooses the next processed operation from the ready pool and sends its descriptor to an appropriate processor agent. After processing this descriptor is placed to the finished operations pool and the information about next stage of processing is changed. The process is repeated while the ready pool is not empty.

For reliability of computations there exists an intermediate working pool of descriptors. This pool is used to mark the descriptors that are now executed by agents. When some agent is broken, then the corresponding descriptor remains in this pool a long period of time. The scheduler periodically checks a descriptor state and moves these waiting descriptors back to the ready pool. The descriptors then have a possibility to allocate on a different working agent.

The agents are dynamically linked up the library of image processing operations. Each processor executes the operations that are specified by the descriptors. The processor receives the descriptor from the coordinator, determines the next operation and executes it using the descriptor data. After a completion of data processing, the descriptor is returned to the scheduler. The processor works while a stop instruction is not received.

Besides the process coordination, the runtime agents check a system state and characteristics. These characteristics are collected and used for a runtime optimization. The optimization is based on a measuring of a data processing speed. When the input data change a system pattern significantly, the system must adapt to this situation. The adaptation consists in a reconfiguration of the operation subsets for all processor agents. The system tries to adapt to changed conditions and to achieve a high processing speed.

The agents use two different policies to choose of next operation from the descriptor pool. The first one consists in choosing operations on the base of

agent's preferences. These preferences are formed in a working process by the means of VAN algorithm. Each agent has a vector of weights, and a probability of a selection of operation *O* for this agent A is:

$$P = \frac{\omega_{O,A}}{\sum_{i=1,O} \omega_{i,A}} \cdot Z(O, A). \quad (1)$$

When the agent performs some operation and its performance characteristics are increased, then the corresponding operation weight is corrected according to:

$$\omega_{O,A}(t+1) = \omega_{O,A}(t) + \alpha, \quad (2)$$

where α is a learning coefficient.

The weights of the remaining operations are corrected according to:

$$\omega_{O,A}(t+1) = \omega_{O,A}(t) - \alpha/(N-1), \quad (3)$$

where *N* means overall amount of the agents. The agent can choose from a subset of operations, taking first ready operation.

The second choosing policy is a greedy one that consists in choosing of first ready operation from the ready pool. This policy is introduced to eliminate a situation, when some descriptors are not chosen by long time. The greedy agents execute these operations and later they can choose this operations type as preferable. Each agent can switch between two scheduling policies randomly.

5. EXAMPLE APPLICATIONS AND EXPERIMENTAL RESULTS

For simulating we use 20×20 μm topological structure (Fig. 5), where black color indicates the exposed part on positive photoresist UV 210 Shipley with a puncture defect.

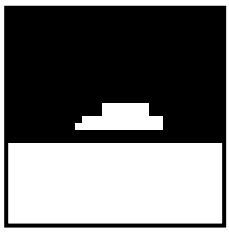

Fig. 5 – Example of topological structure for testing

The parameters for the simulation are: wave-length – 248 nm; generating normalization – 1.0; linear and circular polarization – 0, lens magnification – 0, numerical aperture – 0.6, number of points on the diameter – 19; size of the object – 8.0×8.0 nm, resolution – 0.16, number of points on the object – 50, amplitude transmission – 1.0; index of refraction – 1.0, defocusing within the confines of 1.0-2.0; step defocusing – 0.5.

The results of the simulating are shown in Fig. 6. X-axis is the distance from the center point of the object, the axis Y is the value of the lighting intensity.

Some experiments were conducted to measure a simulation performance depending against the size of the source data and the number of used processors. Checking the result of the simulation was carried out by comparison with the results obtained by leading industry SIGMA C microlithography simulator (Photronics, Inc.) on VLSI layouts with defects according to the standard SEMI-P22-0699 that have been detected EM-6329 [25].

The first group of experiments showed the dependence of the calculation time on the number of processors allocated to run the application (Table 1).

Table 1. The calculation time for different number of processors

Number of objects	Number of processors			
	1	2	3	4
50	77,8	44,67	41,63	24,48
100	154,92	89,23	83,19	49,51
200	310,11	178,47	166,29	101,42

The overall speedup is about 3 times for the case of 4 processors. This result is due to the fact that the parallel application has a nonlinear structure with a different duration of the operations, so its parallelism is also non-linear. Nevertheless, the resulting acceleration can substantially speed up the processing of the photomasks.

The second group of experiments shows the dependence of the performance of parallel applications on the size of the input data (Table 2, the number of objects is equals 100, the number of processors – 4).

Table 2. The calculation time for different data

Number of objects	Number of points of the pupil		
	50	100	200
100	49,51	156,36	534.56

Judging by the results, the algorithm shows almost linear scalability. When the number of points of the pupil increases by 2 times in X and Y coordinates, the amount of data processing is increased by 4 times. In this case, the processing time increases by 3.16 times for 100 pixels, and by 3.42 times for 200 points (about the time for 100 pixels). This demonstrates the scalability of the algorithm, since the processing time is almost linearly dependent on the incoming data volume.

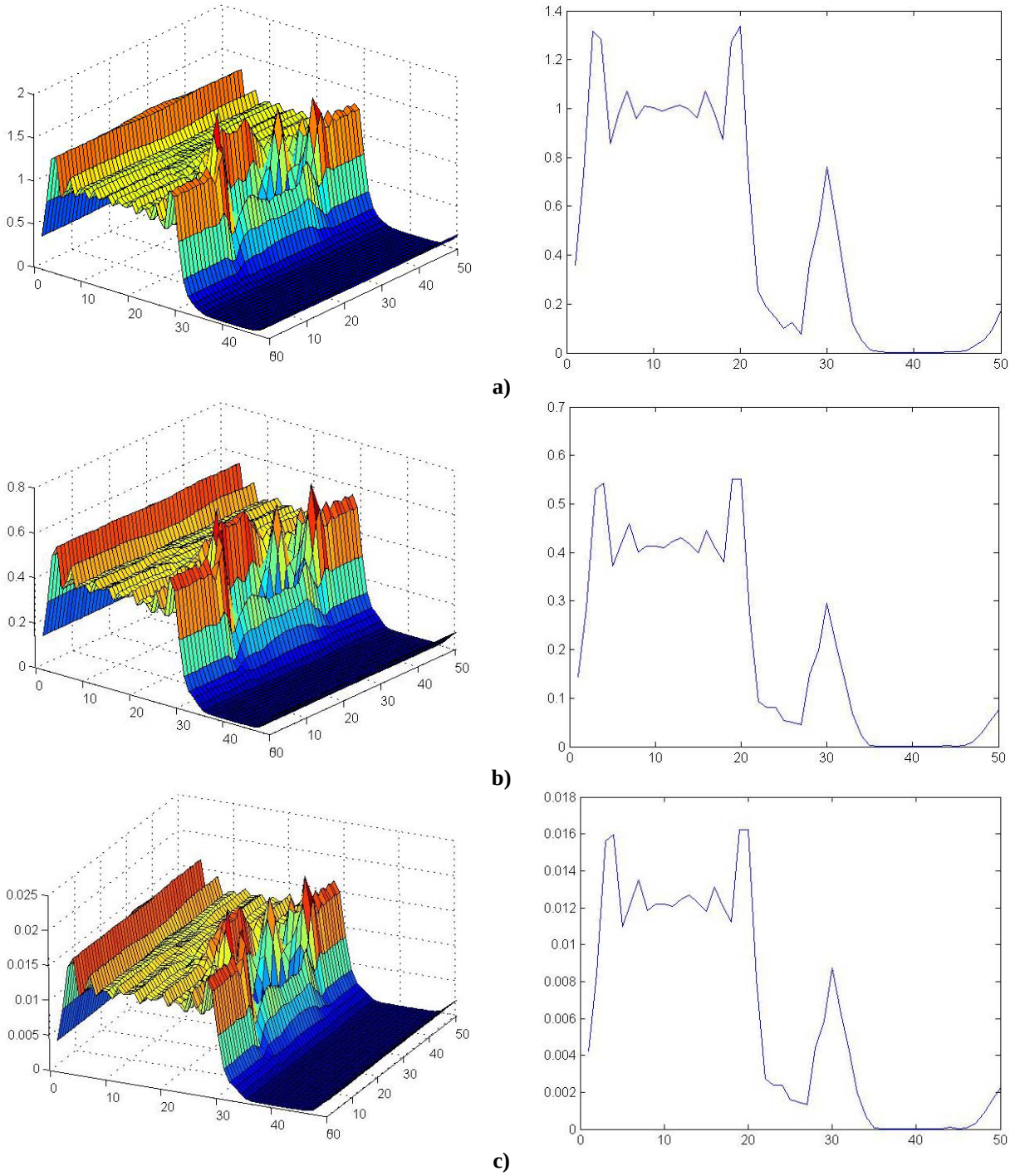

Fig. 6 – Aerial image – intensity distribution on the surface of photoresist (a), past intensity – intensity distribution in the layer of photoresist at the beginning of the exposure (b), absorbed intensity – intensity distribution in the layer of photoresist at the end of exposure (c)

The results of the experiments with the scenarios for calculating the characteristics of aerial images are shown in Table 3 (the processing flow of objects in seconds). It is clear that the optimum time is achieved for 2 processors. Obviously this is due to the large volume of communications between the operations. In this case, a high degree of a locality is required for a quick access to the data.

Table 3. Calculation of the characteristics of aerial pictures

Number of objects	Number of processors			
	1	2	3	4
20	223,21	168,80	178,34	187,47
50	566,99	427,36	442,9	461,27
100	1150,89	865,34	880,74	902,45

The scenario was tested on a sequence of 20 images. The following results of the execution time are presented in Table 4.

Table 4. Calculation of the intensity distribution in the photoresist layer

Number of images	Number of processors			
	1	2	3	4
20	621,35	320,231	263,04	204,36

The results indicate a high degree parallelism of the scenario allowing achieving substantial speed up the processing even for relatively short flows simulated images.

Experimental data flows had an irregular structure and were generated randomly. The results of the static optimization for deterministic flows are presented in the form of comparing the times of stream processing according to the schedules obtained by the classical GA and VAN algorithm. The results are given for different numbers of CPU involved in the processing (Fig. 7).

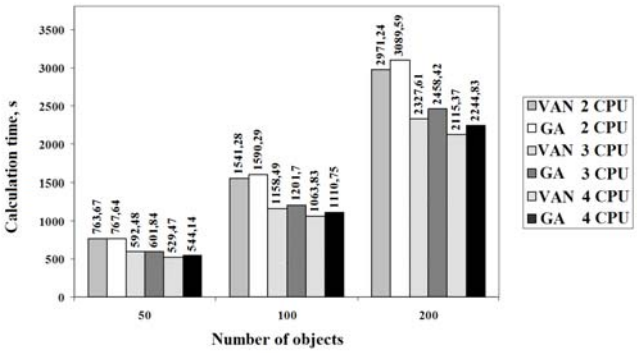

Fig.7 – Improving of a performance for the static optimization

The optimal static schedules, obtained in the first series of the experiments, were used in the second series of the experiments with stochastic flows. These schedules were compared with the schedules which were implemented by dynamically reconfigurable applications (Fig. 8). The results show a change in processing time for the static (S) and dynamic (D) VAN algorithm.

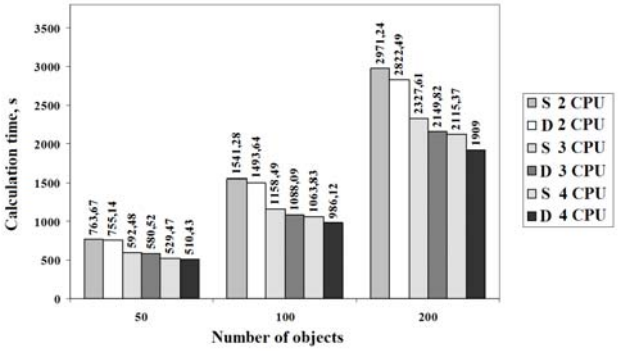

Fig.8 – Improving of a performance for the dynamic optimization

The results of the first series of experiments indicate that VAN algorithm finds the best VAN schedules, and the performance of the algorithm increases with extension of the search space. The VAN algorithm also has a lower computational complexity and finds solutions faster than the classical GA.

The results of the second series of experiments show that the virtual network algorithm significantly improves performance in case of stochastic processing flow of images by operating of dynamic optimization system.

6. CONCLUSION

The implementation of parallel applications in specialized component architectures allows users to significantly accelerate the process of creating, analyzing and optimizing programs. Architecture of data stream processing based on the use of multi-agent systems can be easily adapted for many applications involving parallel processing. Using the component approach defines a flexible framework of a parallel application that allows adapting the computational process for the operation. Design tools can be easily extended with new operations, algorithms, and data types for implementing application processing flow of information from different subject areas.

7. REFERENCES

[1] M. Born, E. Wolf, *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*, seventh ed., Cambridge University Press, 1999.

[2] Y.C. Pati, T. Kailath, Phase-shifting masks for microlithography: automated design and mask requirements, *Journal of the Optical Society of America* A 11 (1994), pp. 2438-2452.

[3] D. Yu, Z. Pan and C.A. Mack, Fast lithography simulation under focus variations for OPC and layout optimizations, *Proc. SPIE 6156*, Apr. 2006, pp. 397-406.

[4] C. Spence, Full-chip lithography simulation and design analysis – how OPC is changing IC design, *Proc. SPIE,* (21) 10 (2005), pp. 1-14.

[5] Eric R. Poortinga, Comparing software and hardware simulation tools on an embedded-attenuated PSM / Eric R. Poortinga [et al.] [Electronic resource]. – 2007. – Mode of access: www.micromagazine.com/archive/00/06/poortinga.html. – Date of access: 12.07.2008.

[6] Optolith – 2D Optical Lithography Simulator [Electronic resource]. – 2005. – Mode of access: http://www.silvaco.com/products/vwf/athena/

optolith/optolith_datasheet.html. – Date of access: 11.07.2008.

[7] N. Cobb and Y. Granik, New concepts in OPC. *Proc. SPIE 5377*, 2004, pp. 680-690.

[8] J. Ye, Y.-W. Lu, Y. Cao, L. Chen, and X. Chen, System and method for lithography simulation, *Patent US 7,117,478* B2, Jan. 18, 2005.

[9] G.A. Gomba, Collaborative innovation: IBM's immersion lithography strategy for 65 nm and 45 nm halfpitch nodes & beyond, *Proc. SPIE 6521*, 2007.

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

[11] 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.

[12] M. Guld, B. Wein, D. Keysers, C. Thies et al., A distributed architecture for content-based image retrieval in medical applications, *Proceedings of the 2nd International Workshop on Pattern Recognition in Information Systems.* 2002, pp. 299-314.

[13] J. Wickel, P. Alvarado, P. Dörfler et al., Axiom – a modular visual object retrieval system, M. Jarke, J. Koehler, and G. Lakemeyer, editors. *Advances in Artificial Intelligence LNAI 2479*. Springer, 2002. p. 253–267.

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

[15] C.J.R. Sheppard, P. Torok, Approximate forms for diffraction integrals in high numerical aperture focusing, *Optik,* (105) 2 (1997), pp. 77-82.

[16] N.B. Voznesensky, A.V. Belozubov, Polarization effects on image quality of optical systems with high numerical apertures, *Proc. SPIE, 1999, Vol.3754*, p.366-373.

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

[18] O. Beaumont, A. Legrand, Y. Robert, Static scheduling strategies for heterogeneous systems, *Computing and Informatics,* (21) (2002), pp. 413-430.

[19] Y.-K. Kwok, I. Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors, *ACM Computing Surveys,* (31) 4 (1999), pp. 406-471.

[20] T. Hagras, J. Janecek, A fast compile-time task scheduling heuristic for homogeneous computing environments, *International Journal of Computers and Their Applications,* (12) 2 (2005), pp. 76-82.

[21] 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), pp. 276-291.

[22] M.A. Palis, J.-C. Liou, and D.S.L. Wei, Task clustering and scheduling for distributed memory parallel architectures, *Trans. Parallel and Dist. Systems*. (7) 1 (1996), pp. 46-55.

[23] 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), pp. 45-71.

[24] Y.M. Yufik, T.B. Sheridan, Virtual networks: new framework for operator modeling and interface optimization in complex supervisory control systems, *Annual Reviews in Control,* (20) (1996), pp. 179-195.

[25] S. Avakaw, A. Korneliuk, A. Tsitko, A prospective modular platform of the mask pattern automatic inspection using die-to-database mask method, *Proc. of SPIE, Photomask and Next-Generation Lithography Mask Technology XII*, Yokohama, Japan, 13-15 April, 2005. – Vol. 5853. – Bellingham, Washington: SPIE, 2005, pp. 965-976.

Syarhei M. Avakaw *Candidate of Sciences degree (an equivalent of Ph.D.) in 2002, Doctor of Engineering in 2008. Director of the KBTEM-OMO (Planar Corporation). Since 1984 has been working on development of the equipment for automatic inspection of planar structures. The main research interests of Dr. S. Avakaw is automatic inspection in precise equipment used for manufacture of high-precision master patterns of the electronics products.*

Alexander A. Doudkin *Candidate of Sciences degree (an equivalent of Ph.D.) in CAD in 1987. Doctor of Engineering in 2010. Principal researcher at the laboratory of systems identification, United Institute of Informatics Problems, NAS of Belarus. Member of Belarusian branch of IAPR and an international society of neural*

networks. Research interests: digital signal and image processing; development of methods, algorithms and technologies for integrated circuit layouts digital processing in respect to creation of computer vision system for VLSI design and manufacture inspection; neural network application, methods of parallel data processing.

Alexander Inyutin *graduated in 1998 from Belarusian State University of Transport BSUT, Gomel, received the MS degree in 1999. He works in laboratory of systems identification, United Institute of Informatics Problems of National Academy of Sciences of Belarus. His research interests include image processing, mathematical morphology, defect detection, methods of parallel data processing.*

Aleksey V. Otwagin *is graduated in Computer Science in BSUIR at 1998. Candidate of Sciences degree (an equivalent of Ph.D.) in 2007. He is associate professor of Belarusian State University of Informatics and Radioelectronics. His research interests include multi-agent systems, parallel processing, optimization of parallel program, genetic algorithms.*

Vladislav A. Rusetsky *is graduated in Computer Science in BSUIR in 2008, received the MS degree in 2009. Now hi is postgraduate at the Department of Electronic Engineering and Technology, BSUIR. His research interests include development of the equipment for photomask production.*