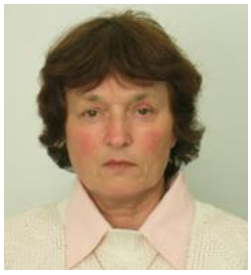
УДК 519.7, 519.17

REMOTE VERIFICATION OF DIGITAL SYSTEMS PROJECTS USING CLUSTER COMPUTERS



D.I. CHEREMISINOV, PhD
Leading Researcher of the Institute of Information Technologies of the National Academy of Sciences of Belarus, Associate Professor

L.D. CHEREMISINOVA, Doctor of Technical Sciences
Chief Scientific Researcher of the Institute of Contemporary Social Sciences of the National Academy of Sciences of Belarus, Associate Professor

United Institute of Informatics Problems, National Academy of Sciences of Belarus, Republic of Belarus
E-mail: {cher, cld}@newman.bas-net.by

Abstract. The paper introduces a methodology that ensures data exchange between the cluster computer and computer-aided design system for remote digital systems verification that is one of the most time-consuming tasks of logical design. The proposed grid service provides the control of parallel computations from the VLSI CAD system environment that makes possible to speed up the solution of tasks arising when performing many design optimization procedures.

Keywords: parallel computing, cluster computer, grid-system, verification, Boolean satisfiability.

Introduction. Almost all of the tasks of logical design are combinatorial in their nature and have an exponential complexity. Combinatorial search algorithms continue to evolve in response to important combinatorial optimization problems that arise within different areas, namely synthesis, optimization and testing of digital circuits [1], mapping, placement and routing of integrated circuits components, artificial intelligence, etc. The distinctive feature of combinatorial problems is that the basic objects for their transformation are logical in their nature (Boolean and *k*-valued vectors and matrices, graphs, Boolean functions, etc.). The deciding processes are quite complex since they often involve the need for searching and analysis of a significant number of variants of intermediate solutions. There is an increasing need for more accuracy and capability from the deciding algorithms, which would involve substantially more computations.

The effective processing of large amounts of computing can be achieved with the use of a parallel processing that is now an attractive solution to reduce the excessive amount of spent time. In recent years, it has become clear that the future of the high performance computing is in cluster computing. That involves the development of parallel algorithms or parallelizing existing sequential algorithms for solving problems. In fact, this involves splitting the process of solving a problem into a number of independent sub-processes that are executed in parallel (but each in sequential way) on different processors of a computer system. This approach works well when the task can be divided according to the data processed by each processor, and the overall solution is made up of partial ones. It is much harder to parallelize combinatorial problem, when simple splitting regarding data is impossible. The problem is how to split the task into subtasks to provide the highest degree of acceleration, which depends on many factors. Moreover, with the same parallelization method, the efficiency

of resulting parallel algorithm largely depends on the task being solved and the architecture of the multiprocessor computer system. Despite a lot of work on this issue, its effective solution has not been currently found.

The paper is devoted to the problem of enhancement of efficiency of computer-aided design (CAD) tools for the sake of computational speedup of underlying combinatorial tasks by means of code parallelization. The goal of the paper is twofold. First one is that to propose a framework simplifying the creation of distributed software for the supercomputer of the family SKIF [2]. We show how the parallel calculations conducted by the supercomputer can be integrated with CAD tools implemented in Windows environment. Since the laborious design task of verification can be converted to a Boolean Satisfiability (SAT) problem [3], as well as many others logic design problems, we propose the technology and results of the remote distributed solution of SAT problem within the environment of software tool of logical design of custom CMOC VLSI [4].

The suggested grid-service is three-layer system consisting of SAT MPI-program for supercomputer, link services and user agent. The grid service is based on using open protocols and middleware.

Parallel Program for SAT Problem Solving. A conjunctive normal form (CNF) formula C is a conjunction of clauses that are disjunctions of one or more literals (being Boolean variables x_i or $\overline{x_i}$). A set of variable assignments satisfies a formula C if all its clauses are satisfied and a clause, in its turn, is satisfied if at least one of its literals is satisfied. The SAT problem consist in finding a satisfying assignment of a given CNF formula (in this case the formula is satisfiable) or determining that such assignment does not exist (the formula is unsatisfiable).

Parallelization of SAT-solvers for a work in a grid environment is a relatively young problem, but it is actively developed in the last years [5], since parallel SAT-solvers often provide faster resolution of SAT instances. There are two main approaches in parallel SAT solving. The first one consists of competitive solvers, or portfolios, in which different computing units solve the same SAT instance in parallel, using different search strategies. The first parallel portfolio SAT was ManySAT [6]. The second approach in parallel SAT solving consists in using cooperative solvers; they work on disjoint subspaces of the search space in parallel. These cooperative solvers require algorithms performing a previous search-space splitting into disjoint subspaces. There are two possible ways of their load distribution: static [7], [8] and dynamic [9] allocation of processor loads. The first approach (opposite to the second one) does not require intensive interprocessor communication, and therefore is well suited for implementation in the grid systems. However, since computing time for these dedicated subspaces cannot be exactly predicted, the hardness of the different SAT instances will be unbalanced, so some of them can verify their formula satisfiability faster than the others.

We use the classical approach to parallelize SAT solving that uses a master-slave method. It is based on splitting the search space such a way that no overlap is possible. The master is responsible for SAT decomposition into small tasks and distribution of these tasks among slave processes as well as for gathering the partial results in order to produce the computation result. The slaves execute a standard cycle: get message with the task, process it and send the result to the master. The communication takes place only between the master and the slaves. Each slave is a SAT solver whereas the master does not participate in the search, but it knows all its slaves and can communicate with them. Slaves do not know each other and cannot communicate directly, only by master. Each slave that is the SAT solver starts with a different fixed partial assignment.

Given CNF formula, the primitive search-space splitting strategy is used to perform a search-space splitting into disjoint subspaces to be explored in parallel. The variables that occur most frequently in clauses are selected as splitting variables to create subtasks. Subtasks are obtained by creating a simple binary search tree based on the possible assignments of the chosen variables. Such a simple way of performing the search-space splitting into disjoint subspaces to be explored in parallel is sufficient since:

1) the time needed for subproblems solution cannot be predicted and so the work cannot be

right balanced prior to the search;

2) the task of search space splitting will be done entirely on the master processor unit in sequential execution mode;

3) it is difficult to find the most relevant set of variables to divide the search space for SAT instance with thousands of variables.

Two approaches of load distribution are used and tested. The first one is static load distribution between computing units that is done prior to the search process and is not rearranged during the computing process. In that case, the analyzed Boolean formula is divided on as many subformulas as the number of available cluster computing nodes. And each of the subformula is processed in a separate cluster computing unit in parallel up to the end. The master is responsible for decomposition of the problem into small tasks and distribution of these tasks among slave processes as well as for gathering the partial results in order to produce the final result of the computation. The slaves execute a very simple cycle: get message with the task, process it and send the result to the master. The communication takes place only between the master and the slaves.

When a slave process stops working, it sets: 1) a ternary vector t specifying that the desired satisfying assignment is found, or 2) NULL vector that means the analyzed minor is unsatisfiable. In the first case the SAT problem is solved and the given CNF formula is satisfiable. The master unit sends message to slaves to stop working indicating that the problem has been solved. In the second case, the slave does not find a solution and it becomes idle. The master verifies whether the pool of minors is not empty. If yes, it retrieves one of the minor vectors and provides it to the idle slave as a new search space.

The described parallel method for SAT solving is implemented by MPI-program «Calculator» for a cluster computer. The program source data is CNF description in the text or in DIMACS [10] formats that is a standard interface for SAT-solvers and provides a compact representation of rare ternary matrixes, which are obtained by formal electronic circuit verification. The program «Calculator» output is the text message containing the mode and duration of the solution, as well as satisfying vector if the tested CNF is satisfiable.

The Architecture of the Distributed Software for Cluster Computer. The supercomputer of SKIF family [2], as well as majority of clustered computers, has Beowulf architecture. It consists of large number of computing nodes of the same type that work together so they can be viewed as a single computing system with the Linux-similar operating system and MPI (Message Passing Interface) [11] implementation installed to allow programs to be run across all nodes. The cluster nodes are used only for the execution of the code and to exchange data between processes running on them using MPI. MPI-executable code of a program is copied to the memory of each node in the cluster that is assigned by control machine for its implementation. All nodes start to operate simultaneously in parallel.

To start tasks, the Portable Batch System (PBS) [12] is running on the cluster control machine. Its primary task is to allocate computational tasks among the available computing resources. To run MPI-program in batch mode, a control script (batch file) should be submitted to the PBS system. It contains a set of commands which user can manually enter in the command-line. Particularly, the script sets the program to run and required resources (number of processes, run time, etc.). To communicate with the environment, OpenSSH server [13] (open the secure shell) is installed in control machine providing the most common way of connection to a remote Linux server through SSH protocol. Using OpenSSH, the remote user can interact with cluster control machine.

The suggested grid service for solving systems of logic equations for the cluster computer has a typical structure of the grid systems for high-performance computing and it is considered as a three-layer system consisting of a MPI-program «Calculator» for a supercomputer (the main component of the system), link services and the user agent.

The user agent interacts with the «Calculator» through the communication service and can manage the grid service via an agent, such as a web browser, or directly from CAD VLSI. In distributed

systems, a service is a program running without user interaction, whose purpose is to serve other applications, which can be user agents or other services. Service connection creates a link to a supercomputer by standard, open, universal protocols and interfaces (Fig. 1).

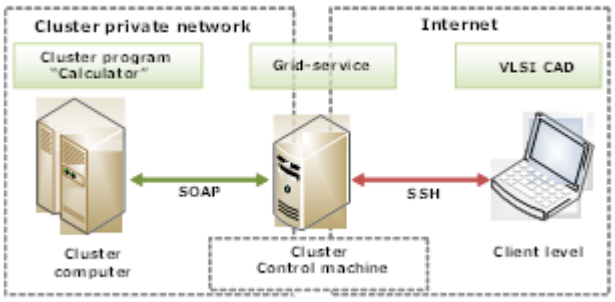


Figure 1. Distributed program of grid service for logic equation solving by the supercomputer

The grid-service is built on the basis of generally accepted open standards and cross-platform middleware, which is the agent between programs running on remote computers. The communication service (the grid service middleware) is based on the technology of Web-services integration with Simple Object Access Protocol (SOAP) – a protocol for exchanging structured messages in a distributed computing environment. SOAP is based on XML and expands the application layer protocol. Agents interacting with this protocol, have a simple behavior: request/response. SOAP sender agent sends the XML-message by HTTP-request and gets the result as the HTTP-response from the recipient agent. SOAP protocol creates communication link between the job management system of the cluster computer and the communication server with the user's client (Fig. 1). The protocol of the grid service consists of the following steps (Fig. 2.): 1) convey the file with CNF, which should be analyzed, to the to file system on the cluster control machine; 2) start the program «Calculator» to solve the system of logic equations on the supercomputer; 3) return the file with the obtained results to the client computer.

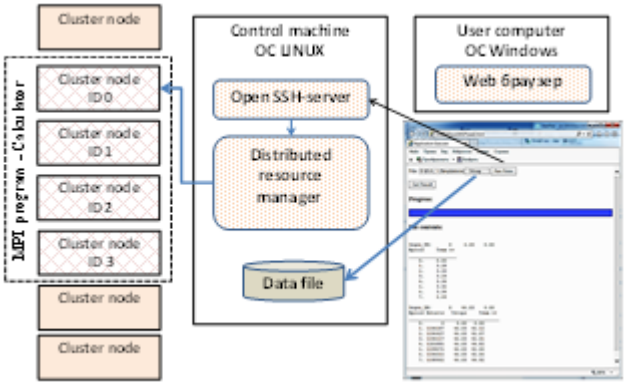


Figure 2. The architecture of the logic equation solving service

The client part of the Grid service can interact with the program on the supercomputer through Middleware UNICORE [14], which is an open software package compatible with standard protocols. Three-layer architecture of UNICORE includes: a user level, UNICORE server and supercomputer. UNICORE interface is focused on batch job processing by supercomputer and uses an object-oriented approach. The batch job script specifies which data should be imported to the cluster's file system before program processing and what needs to be exported after the job is completed. These import and export operations are performed by the UNICORE interface. The cluster supercomputer in the

UNICORE model is presented as a virtual site with a single file space. Each site consists of an interface to the local operating system and a batch job management system.

The technique of launching the programs on the cluster computer for solving combinatorial problems arising in the process of integrated circuits design was worked out on the example of the "Energy-saving logical synthesis" software system (ELS) [4] that was developed in the laboratory of the logical design of the United Institute of Informatics Problems of the National Academy of Sciences of Belarus. The ELS system is designed to automate the design of multi-level logic circuits on basis of the library elements of custom CMOS VLSI with the area layout and energy consumption optimization.

Integration of Software Module for SAT Solving in the VLSI Design Route. One of the key design stages within the framework of the ELS software tool is the verification of design solutions at all design stages [15]. The verification program works for any pairs of project states (functional or structural descriptions) that can belong to the same project (or different). In the framework of the ELS system, verification procedures are implemented, that are based on the simulation of the descriptions of the verified device (on the area of its specification only) and a formal proof of the functional identity (or implementability, in case of descriptions with functional uncertainty) of projects or based on hybrid approaches.

Formal verification is based on reducing the problem to verifying the satisfiability of some CNF *C*, which specifies the structure of the compared descriptions. The satisfiability of the CNF indicates the violation of the equivalence of the compared descriptions. As a rule, the CNF being analyzed for satisfiability has a significant size, which affects the speed of the verification program. That is why it was decided to solve this task with the help of the cluster computer.

Since the set of project operations in the ELS system is fixed at the user interface level, the verification program has been modified to integrate the grid service of solving logical equations within it. After receiving the CNF in the process of performing the verification operation in the environment of the ELS system, the following actions are performed:

1) instead of calling the internal program to solve logical equations (SAT Solver Minisat), the file with the CNF, which should be analyzed, is specified in the DIMACS format;

2) the file with the CNF is conveyed to the cluster control machine;

3) the program «Calculator» is started on the cluster computer;

4) then, after finishing the program «Calculator», the file with the solution results returns to the ELS system.

If the returned file contains the found satisfying vector (CNF is satisfiable), it is concluded that the compared descriptions are not equivalent, otherwise the descriptions are equivalent.

In order to test the proposed approach to the distributed solving the verification problem, a number of structural descriptions, which arise during the design process in the environment of the software complex ELS, have been verified. The following example can be used to show the complexity of the tasks to be solved: the verified circuits consisted of 1923 CMOS library elements; the conventional CNF in this case depended on 13103 variables and contained 38635 clauses. The general limitation of the «Calculator» program to the maximum size of the analyzed CNF is about 109 clauses.

In addition, in order to determine the efficiency of the «Calculator» program on a supercomputer, it was subjected to tests on a stream of random CNFs with the specified parameters and on several CNFs [10], which were used to test SAT-solvers in international competitions of these programs and for which the results of their solutions are known. The results obtained with "Calculator" coincided with the known solutions.

To evaluate the efficiency of solving the CNF satisfiability problem with the use of a supercomputer, the acceleration coefficient $S = t(1)/t(n)$ was adopted, where $t(1)$ is the time spent to solve the problem by a sequential algorithm on one processor, and $t(n)$ is the time of solving the same problem by a parallel algorithm using n processors of a cluster computer system. For the case $S = l$, a linear acceleration is obtained, which is proportional to the number of processors. The behavior of

the «Calculator» strongly depends on the problem being solved. If CNF is satisfiable, then «superlinear» acceleration is possible [16]. If the CNF is not satisfiable, the acceleration corresponds to the Amdahl law and it is less than linear [16], since each of the subordinate processors must examine the entire its subspace to the end to prove the absence of a solution.

References

[1]. Zakrevskij, A.D. Logicheskie Osnovy Proektirovaniya Diskretnykh Ustroistv / A.D. Zakrevskij, Yu.V. Pottosin, L.D. Cheremisinova. – Moscow: Phizmatlit, 2007, 589 p. (in Russian).

[2]. Abramov, S.M., Principles of Construction of Supercomputer family «SKIF» / S.M. Abramov, N.N. Paramonov, V.V. Anistchenko, S.V. Ablamejko // Informatika. – 2004. – № 1. – С. 89–106 (in Russian).

[3]. Ganai, M. SAT-Based Scalable Formal Verification Solutions / M. Ganai, A. Gupta. – New York: Springer-Verlag, 2007. – 338 p.

[4]. Bibilo, P.N. Low-Power Logical Synthesis of CMOS Circuits Automation / P.N. Bibilo, L.D. Cheremisinova, S.N. Kardash, et al. // Programnaia indzeneria, 2013, № 8, с. 35–41.

[5]. Hyvarinen, A.E.J. Partitioning SAT Instances for Distributed Solving / A.E.J. Hyvarinen, T.A. Junttila, I. Niemela // Lecture Notes in Computer Science. Springer, Heidelberg. – 2010. – Vol. 6397. – P. 372–386.

[6]. Hamadi, Y. ManySAT: a parallel sat solver / Y. Hamadi, S. Jabbour, L. Sais // Journal on Satisfiability, Boolean Modeling and Computation. – 2009. –No 6. – P. 245–262.

[7]. Schubert, T. PaMiraXT: Parallel SAT Solving with Threads and Message Passing / T. Schubert, M. Lewis, B. Becker // Journal on Satisfiability, Boolean Modeling and Computation. – 2009. – Vol. 6. – P. 203–222.

[8]. Toropov, N.R. A Parallel Tautology DNF Checking / N.R. Toropov // Informatika. – 2005. – № 2. – С. 35–42 (in Russian).

[9]. Hyvarinen, A. Grid-Based SAT Solving with Iterative Partitioning and Clause Learning / A. Hyvarinen, I. Niemela, T. Junttila // LNCS, 2011. – Vol. 6876. – P. 385–399.

[10]. CNF Files. Available at: http://people.sc.fsu.edu/~jburkardt/ data/cnf/cnf.html (date of access: 10.02.2017).

[11]. Message Passing Interface Forum. MPI: A Message-Passing Interface standard, version 1.1. Available at: http://www.mpi-forum.org/docs (date of access: 10.02.2017).

[12]. PBS (Portable Batch System) Basics for UNIX. Available at: http://tx.technion.ac.il/usg/PBS/PBS_basics.pdf (date of access: 10.02.2017).

[13]. OpenSSH Server. Available at: https://help.ubuntu.com/ lts/serverguide/openssh-server.html (date of access: 10.02.2017) (in Russian).

[14]. Operational Center of National grid-system of Republik of Belarus. Available at: http://noc.grid.by/index.php?n=Main. Download (date of access: 10.02.2017) (in Russian).

[15]. Cheremisinova, L.D. Software tools for verification of descriptions of combinational circuits during the process of logic design / L.D. Cheremisinova, D.Ya. Novikov // Programnaia indzeneria. – 2013. – № 7. – С. 8–15 (in Russian).

[16]. Cheremisinov, D.I. Design and the analysis of parallelism in processes and programs / D.I. Cheremisinov. – Minsk: Belaruskaja navuka, 2011. – 300 p. (in Russian).