ISSN 0361-7688, Programming and Computer Software, 2011, Vol. 37, No. 3, pp. 147–152. © Pleiades Publishing, Ltd., 2011.
Original Russian Text © A.S. Kamkin, M.M. Chupilko, 2011, published in Programmirovanie, 2011, Vol. 37, No. 3.

Survey of Modern Technologies of Simulation-Based Verification of Hardware

A. S. Kamkin and M. M. Chupilko

Institute for System Programming, Russian Academy of Sciences, ul. Solzhenitsyna 25, Moscow, 109004 Russia
e-mail: kamkin@ispras.ru; chupilko@ispras.ru

Received September 15, 2010

Abstract—Comparative analysis of modern approaches to simulation-based verification (testing) of hardware models—AVM (advanced verification methodology) developed by the Mentor Graphics Corporation, OVM (open verification methodology) developed jointly by Mentor Graphics and Cadence Design Systems, and the UniTESK (unified testing and specification tool kit) technology developed at the Institute for System Programming of Russian Academy of Sciences—is given. In the paper, advantages and disadvantages of these approaches are analyzed, and architectures of the test systems are compared. Recommendations on the development of the UniTESK technology and its unification with the OVM methodology, which is currently widely used and pretends to be a standard in the field of hardware verification, are given.

DOI: 10.1134/S0361768811030017

1. INTRODUCTION

The growth of hardware complexity inevitably results in the development of technologies of test system design intended for simulation-based verification. Currently, the level of the hardware complexity is so high that testing cannot be carried out without applying automation methods. Such methods make it possible to develop test systems that automatically generate test sequences, estimate correctness of system behavior, and collect information on the achieved level of test coverage. However, one should clearly understand that the automation is only a part of the modern technology of test development.

Under the condition of permanent modifications of requirements and project redefinitions, the *possibility of tests reuse and possibility of creation of tests stable to variations of the implementation* become very important. The existing approaches to the construction of test systems, such as AVM (advanced verification methodology), URM (Universal Reuse Methodology), OVM (open verification methodology), and UniTESK (unified testing and specification toolkit), to some extent, solve the above-indicated problems; however, the diversity of the existing approaches hampers interaction between different groups of designers.

Standardization of the test development technologies is a necessary step in the development of the semiconductor industry. The appearance of standards creates prerequisites for independent development of tests, which positively affects the market of the ready-to-use components (IPs, Intellectual Property Cores). If tests for a hardware component are developed in the framework of a commonly accepted approach, then third engineers will be able to understand and, if required, to complement them.

Recently, the OVM technologies developed jointly by the Mentor Graphics and Cadence Design Systems Corporations are widely used in testing hardware models. Predecessors of OVM are AVM by Mentor Graphics and URM developed by Cadence Design Systems. In this paper, the OVM technology, together with its predecessors, is compared with the UniTESK technology developed at the Institute for System Programming of Russian Academy of Sciences. In the paper, advantages and disadvantages of these approaches are analyzed, architectures of the test systems are compared, and recommendations on the development of the UniTESK tools and its unification with the OVM approach are given.

The paper consists of seven sections, including the Introduction and Conclusions. In Section 2, basic concepts used in various technologies are introduced. The third, fourth, and fifth sections are devoted to surveys of the AVM, OVM, and UniTESK technologies, respectively. In these sections, basic principles of these approaches, and architectures of the test systems are discussed. Comparative analysis of the technologies discussed is given in Section 6. Finally, in Section 7, the conclusions are presented, and recommendations on the development of the UniTESK technology are given.

2. BASIC CONCEPTS

In this paper, under *hardware models*, we mean program models of digital devices described in special

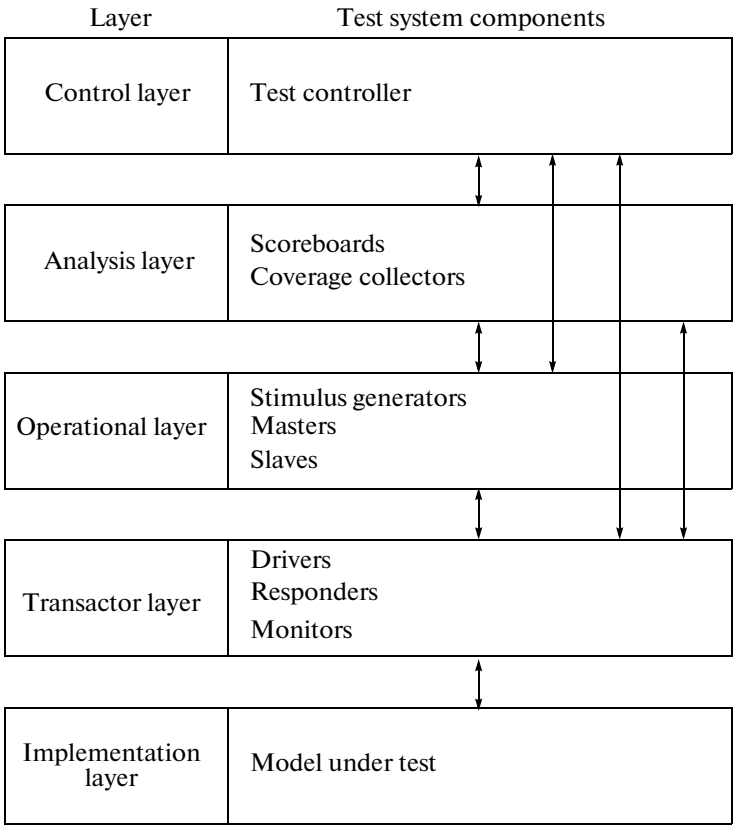

Fig. 1. Layers of the AVM test system.

languages. Such languages can be divided into two groups:

– *hardware description languages (HDLs)*, by means of which *register transfer level (RTL) models* are developed; the examples of such languages are Verilog [1] and VHDL [2];

– *system-level design languages*, which describe hardware systems on a higher level of abstraction and support development of the software part of the system; the examples are SystemC [3] and System Verilog [4].

In what follows, for convenience, we will refer to languages of both kinds as *hardware description languages*, or *HDL languages.*

The technologies discussed in the paper are aimed at the automation of the development of tests for hardware models and creation of test systems capable of multiple reuses and stable to modifications of the implementation.

Before we start to describe the AVM, OVM, and UniTESK technologies (the URM technology mentioned in the Introduction is not described here in view of complete lack of documentation on it), we note the following. The names of the AVM and OVM approaches contain word *"methodology,"* whereas UniTESK is considered to be *"technology"* from the point of view of its developers. In this paper, we do not distinguish between these two terms; the both terms are meant to denote a collection of principles, methods, and tools for the development of test suites.

3. SURVEY OF THE AVM TECHNOLOGY

The advanced verification methodology (AVM) was developed by the Mentor Graphics company and is distributed under the Apache 2.0 license. Currently, it is not standardized, and all information about it can be found in the cookbook [5] and the user guide.

3.1. Basic Principles of AVM

By means of the AVM technology, complex test systems in languages SystemVerilog and SystemC are developed. The basic means of AVM are libraries of classes developed in these languages. In the development of test systems, the following basic principles are used: *object-oriented approach, transaction-level modeling (TLM), and constraint-random generation.*

TLM is an approach to modeling of data transactions in digital systems in the framework of which organization of links between system modules is separated from their implementation. A *transaction* is meant to be a single transfer of control information and data between two modules. For example, in the

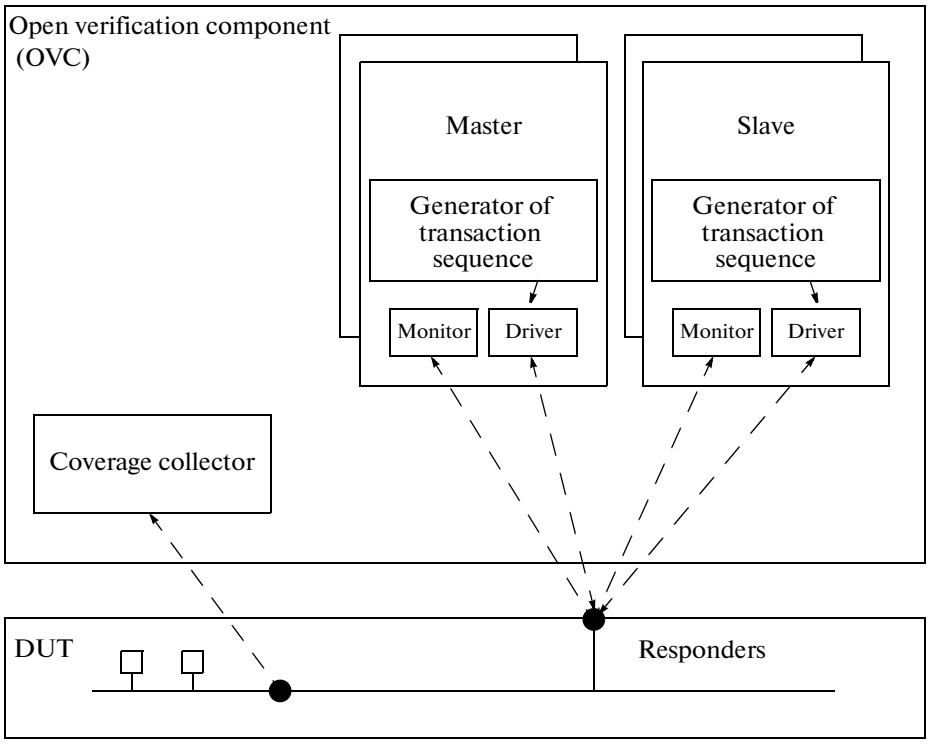

Fig. 2. Open verification component.

projects where buses are used for data transfers, operations of reading from and writing to a bus can be described in terms of transactions; in systems with commutation of packages, a transaction is an operation of sending or receiving a package.

The use of object-oriented approach and transaction-level modeling allows the AVM technology to achieve high rate of multiple test reuses. The use of mechanisms of class inheritance and method reload makes it possible to redefine behavior of a test system component. Owing to this, the time spent for the test development and debugging reduces considerably.

3.2. Architecture of the AVM Test System

Test systems created with the use of the AVM technology should be partitioned into several layers (Fig. 1). The lowest level is the RTL model of the design under test (DUT). The second level is that of *transactors*, i.e., components of the test system that transform a flow of input transactions to input signals of the model under test and, vice versa, transform output signals of the model under test into a flow of output transactions.

Examples of transactors are *driver* (converts the flow of transactions to input signals of the model under test), *responder* (responds to requests of the model under test on supplying additional data), and *monitor* (monitors outputs of the model under test not affecting its behavior).

The next level is the so-called *operational layer*. Components of this level form the test environment required for the model under test and are as follows:

1. The *stimulus generator* creates a flow of input transactions (stimuli). To generate stimuli, an approach based on the constraint-random verification (CRV) is used. The generator communicates with the model under test via a driver.

2. A *master* is an element of the environment of the model under test that initiates interaction with it. The master sends sequences of (random or aimed at achieving a certain situation) transactions and may use a feedback to determine its subsequent actions. The master communicates with the model under test via a driver.

3. A *slave*, like the master, models the environment of the model under test; however, unlike the latter, it is a passive component responding to requests of the model.

The next level is the *analysis level*, the components of which check correctness of the model behavior, estimate testing completeness, and perform some other functions. The basic components of this level are a *scoreboard* (checks correctness of the model behavior in response to the input transactions) and *coverage collector* (checks testing completeness by calculating the numbers of events of specified types occurring in the course of testing).

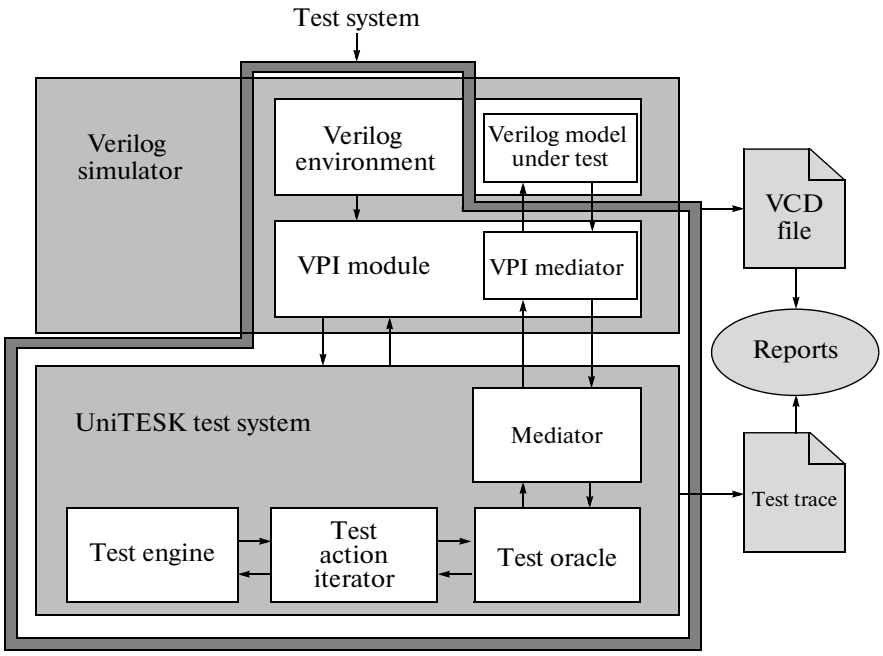

Fig. 3. Architecture of the UniTESK test system.

The uppermost *control level* includes only one component *test controller*, which coordinates work of other components of the system under test.

Figure 1 illustrates the multilayer structure of the test system. Each rectangle in the figure corresponds to a certain layer. The name of the layer is shown on the left, and the basic components of the layer are presented on the right. The arrows show interactions between the layers.

4. SURVEY OF THE OVM TECHNOLOGY

The open verification methodology (OVM) was created jointly by the Mentor Graphics and Cadence Design Systems corporations. OVM is the first *open* industry method of test development based on the SystemVerilog language [6]. Predecessors of OVM are AVM (Mentor Graphics) and URM (Cadence Design Systems). The OVM technology is not standardized by now, but it is considered to be the first candidate to become the standard.

4.1. Basic Principles of OVM

Basic principles of the OVM technology are similar to those of AVM. The basic means of OVM is a library of classes in the SystemVerilog language (it is planned to include support of SystemC as well), which allows one to create modular test systems that can multiply be reused. One of the advantages of this technology is *coverage-driven verification (CDV)*, which makes it possible to clearly identify goals of the testing and to better control the process of the test development.

4.2. Architecture of the OVM Test System

In the framework of the OVM technology, the test system is assembled from special blocks called *open verification components (OVCs)*. Each block encapsulates test environment for a separate hardware module and consists of a complete set of components ensuring stimulus generation, verification of behavior correctness, and estimation of testing completeness. If a hardware model under test is obtained by integrating several modules, the test system for it is constructed by combining the corresponding OVC blocks and constructing a virtual sequencer over them.

Structure of the OVC blocks is almost identical to the architecture of the AVM test system. An example of an open verification component is depicted in Fig. 2.

5. SURVEY OF THE UNITESK TECHNOLOGY

The unified specification and testing tool kit (UniTESK) [7] has been developed at the Institute for System Programming of Russian Academy of Sciences. Unlike the above-discussed AVM and OVM technologies, which were created in commercial organizations, UniTESK is an academic development, which, nevertheless, relies on a real experience of software and hardware testing.

5.1. Basic Principles of AVM

In the development of test systems by the UniTESK technology, the following basic principles are used:

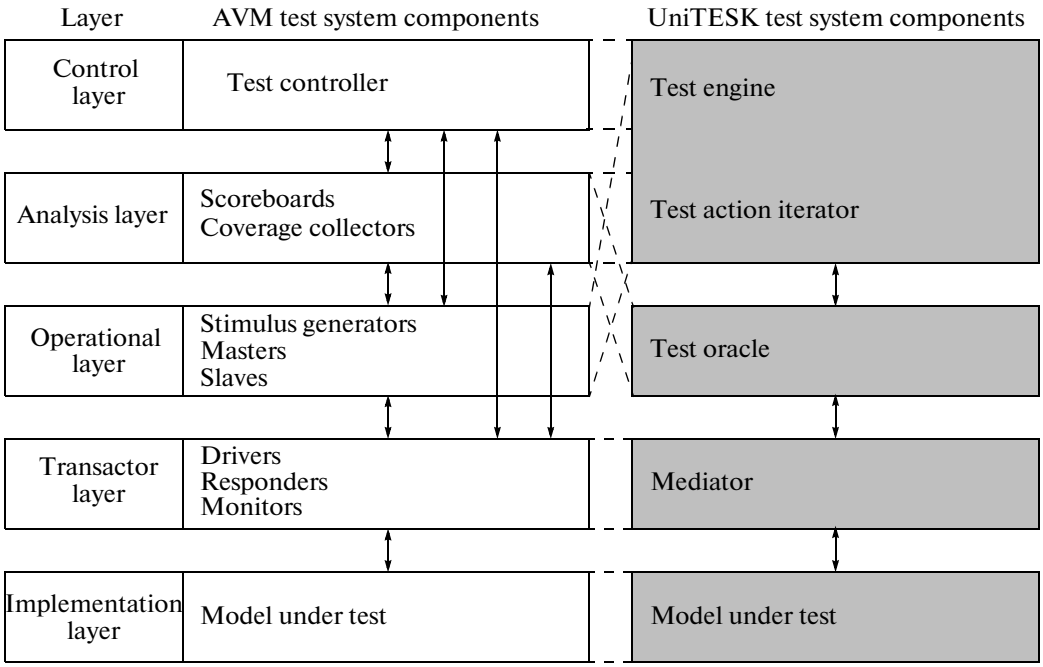

Fig. 4. Comparison of architectures of the test systems.

—definiteness of the set of components with explicit separation of functions and clear interfaces;

—use of formal specifications in the form of pre- and postconditions of interface operations and data type invariants for automated generation of test oracles (verification components);

—use of finite automaton models for constructing sequences of test actions (stimuli).

The basic means for testing hardware models is the CTESK tool, which uses language SeC (specification extension of C) for the development of the test system.

5.2. Architecture of the UniTESK Test System

The UniTESK test system consists of the following basic components (Fig. 3):

1. The *test engine* is a library component of the test system. The test engine is based on the algorithm for traversing the state graph of a finite automaton that models the target system at a certain abstraction level.

2. The *test action iterator* works under control of the test engine and is designed for searching for admissible test actions in each achievable state of the finite automaton. The iterator of test actions is generated automatically from the test scenario, which is an implicit description of the finite-automaton model of the system under test.

3. The *test oracle* estimates correctness of the target system behavior in response to a single test action. It is generated automatically from the formal specifications describing requirements to the target system in the form of pre- and postconditions of the interface operations and data type invariants.

4. The *mediator* relates formal specifications to a particular implementation of the system under test. The mediator transforms a single test action from the specification representation to the implementation representation and the response obtained from the implementation representation to the specification representation. In addition, the mediator synchronizes the state of the specification with the state of the system under test.

5. The *test trace* reflects events occurring in the course of testing. Based on the trace, it is possible to automatically generate various reports, which help to analyze testing results.

In addition to basic components, the test system includes parts responsible for the integration with a particular HDL language. For example, for the Verilog language, such components are *VPI*[1] *module* (including *VPI mediator*) and the *Verilog environment.*

6. COMPARISON OF THE TEST DEVELOPMENT TECHNOLOGIES

Let us compare the AVM, OVM, and UniTESK technologies. The AVM and OVM approaches have been developed by commercial organizations with regard to the great experience of hardware models

[1] VPI (Verilog procedural interface) is a standard interface designed for calling functions written in C and other programming languages from Verilog modules.

testing accumulated during recent years. The basic stress in these technologies is made on multiple test reuses, which is achieved by means of the object-oriented programming and transaction-level modeling. Besides, the OVM approach relies on a mechanism of construction of test systems from ready-to-use blocks responsible for testing separate modules of the hardware system. This concept reflects trends in modern industry, when the basic work is concentrated on the integration of the hardware from ready-to-use modules rather than on the design of fundamentally new hardware.

In the UniTESK technology, which is an academic development, the main stress is made on the automation of the test creation. In the UniTESK, formal behavior specifications in the form of pre- and postconditions and data type invariants, as well as finite-automaton models for constructing stimuli sequences, are used. All this makes it possible to develop high-quality tests that carefully check functionality of the model under test. Among the advantages of UniTESK, we note the possibility of cycle-accurate specification development based on the pre- and postconditions of the operation stages. As for the means of multiple test reuse, they are less advanced in the UniTESK technology compared to AVM and OVM. This is explained mainly by the fact that the CTESK tool used for testing hardware models is based on the C language. Implementation of the basic principles of UniTESK for the SystemVerilog and SystemC languages would make it possible to improve characteristics of the test reuse.

Comparison of architectures of the test systems AVM (it is the same as that of the OVC block) and UniTESK is shown in Fig. 4.

The transactor layer in AVM corresponds approximately to the mediator layer in UniTESK. The correspondence between the upper layers is more difficult to establish. The AVM stimulus generator corresponds to the UniTESK test action iterator (together with the test engine), and scoreboards, to the test oracle. The UniTESK component that is most related to the AVM test controller is the test engine. Note that some AVM components have no analogues in the UniTESK technology: for example, no components similar to the masters or slaves are available in UniTESK.

7. CONCLUSIONS

We think that there should appear standards in the field of test development for hardware models shortly. The main candidate for the role of the standard is the OVM technology, which is promoted by such large companies as Mentors Graphics and Cadence Design Systems. This technology is based on the well-accepted approaches AVM and URM and is actively used in industry. We advocate the unification of the UniTESK technology (as well as other methods of automation of test development for hardware models) with the OVM approach. It is better to use one approach, which can be understood by different groups of designers, than several ones.

Under the unification, we mean the following. The basic principles underlying the UniTESK technology, first of all the use of formal specifications in the form of pre- and postconditions for description of the system and the use of finite-automaton models for generation of test sequences, performed very well in practice. Nothing prevents from the implementation of these principles in terms of the OVM technology based on a library of base classes. This will make it possible to create tests by the UniTESK technology that will acquire new quality: they can be reused in construction of larger test systems, and the like.

REFERENCES

1. IEEE Standard VHDL Language Reference Manual, *IEEE Std 1076–1987*.
2. IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language, *IEEE Std 1364–1995*.
3. IEEE Standard SystemC Language Reference Manual, *IEEE Std 1666–2005*.
4. IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language, *IEEE Std 1800–2005*.
5. Rose, A., Fitzpatrick, T., Rich, D., and Foster, H., *Advanced Verification Methodology Cookbook*, Mentor Graphics Corporation, 2008. http://www.mentor.com/go/cookbook
6. *OVM Whitepaper*, Mentor Graphics Corporation, Cadence, 2007. http://www.ovmworld.org
7. Ivannikov, V.P., Kamkin, A.S., Kuliamin, V.V., and Petrenko, A.K., Applying UniTESK Technology to Functional Testing of Hardware Models, *Preprint of Inst. of System Programming RAN*, Moscow, 2005 http://citforum.ru/SE/testing/unitesk_hard/.