Teaching Digital RT-Level Self-Test using a Java Applet

S. Devadze, A. Jutman, A. Sudnitson, R. Ubar
Tallinn Technical University,
Raja 15, 12618 Tallinn, Estonia,
raiub@pld.ttu.ee

H.-D.Wuttke
Technical University Ilmenau, Germany
Dieter.Wuttke@theoinf.tu-ilmenau.de

Abstract. *A method called "living pictures" to combine learning, training and research phases in a laboratory course for educating today's VLSI and system designers is presented and implemented as an Internet based tool. The system is designed mainly to illustrate RT-level (RT-Register Transfer) problems in control intensive digital systems including: investigation of tradeoffs between system's speed and HW cost, RT-level simulation, fault simulation, test generation, built-in self-test (BIST) and others. In this article we concentrate mainly on testing related problems. The described system has a built-in multilingual support to ensure easy integration into teaching courses of universities over the world.*

1. INTRODUCTION

Advances in the areas of deep-submicron electron technology and design automation tools are enabling engineers to design larger and more complex integrated circuits, and driving them toward new System on a Chip (SOC) design methodologies. SOC is seen as a major new technology and the future direction for the semiconductor industry. The more complex electronics systems are getting, the more important will be the problems of testing and design for testability. The costs of test is becoming the major component of the manufacturing cost of a new product. Today, design and test are no longer separate issues. The emphasis on the quality of the shipped products, coupled with the growing complexity of systems design, require testing issues to be considered early in the design process.

Recent reviews have discovered that most VLSI and system designers know little about testing because of the gap in education [1]. Today's university courses on design very seldom handle the topics of testing in sufficient details. Entering into SOC era means that test must become an integral part of the VLSI and system design courses. The next generation of engineers involved with VLSI technology should be made aware of the importance of test, and trained in test technology to enable them to produce high quality, defect-free products.

Design for Testability (DFT) is rapidly becoming one of the key considerations in today's SOC designs. Moving towards multi-million gate SOCs makes embedded testing strategies via Built-In Self-Test (BIST) architectures mandatory. It is critical to ensure that students will be equipped with skills in DFT and BIST, and get hands-on experience in solving test problems in digital systems to make them successful designers when they leave university [2].

In the following a conception and tools are presented to increase the teaching quality in the field of electronics design and test. Both, gate level and RT-level testing problems are covered. To illustrate design and test problems together, we use the same environment. The system supports the possibility of distance learning as well as a web-based computer-aided teaching. The interactive modules are focused on easy action and reaction, learning by doing, a game-like use, and encourage students for critical thinking, problem solving, and creativity.

The paper is organized as follows. Section 2 gives an overview of the new teaching concept supported by the paper. In Section 3 the simulation environment is described. Section 4 presents the concepts of teaching high-level test topics and in Section 5 conclusions are made.

2. SYSTEM OVERVIEW

The core of the teaching concept presented here is a Java-applet of a special type, which we call "Living Pictures" [3]. Those applets simulate tricky, quite complicated situations of the learning subject in a graphical form on the computer screen. The graphics is self-explanatory and provides interaction possibilities. By using these possibilities the students can generate examples that are interesting enough to encourage their own experiments but not too complicated for learning.

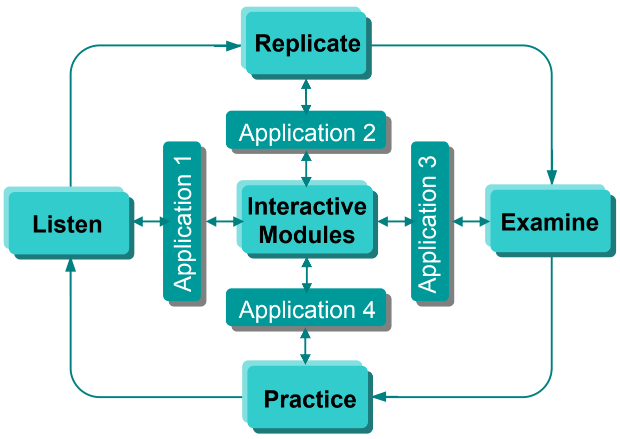


Fig. 1 The four phases of learning process

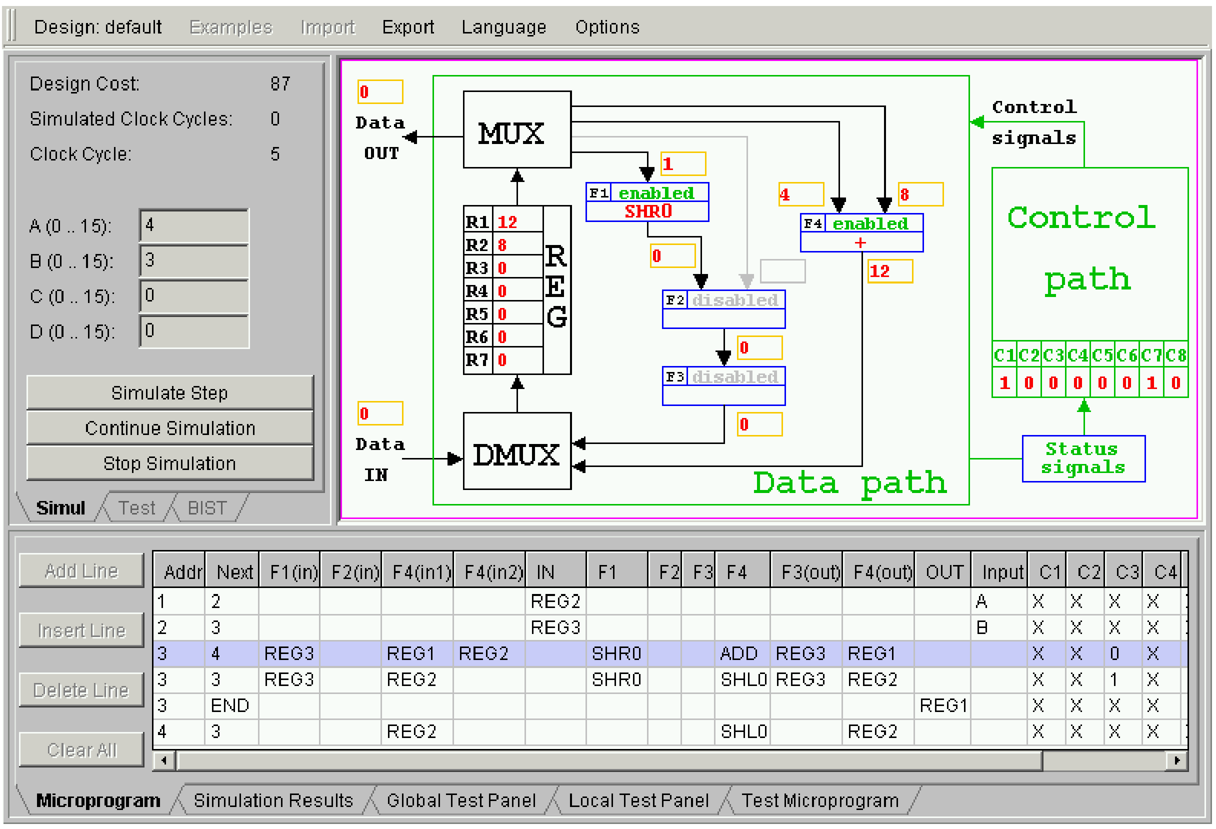
Fig.1 shows the four phases of the learning process supported by the education system: listening, replication, examination and practice phase. The system supports the action based training since for each phase there exists a special application service adapted to the learning process which allows different views and actions using the same interactive module.

In our teaching system [4] we combine and illustrate many different problems related to both RT-level control intensive digital design and test. This gives a unique possibility to teach all of them in a consecutive iterative approach. The range of taught problems includes:

- design of a data path and control path (microprogram) on RT level
- investigation of tradeoffs between speed of the system & HW cost
- RT-level simulation and validation
- gate-level deterministic test generation and functional testing
- fault simulation
- logic BIST, circular BIST, functional BIST, etc.
- design for testability

The teaching system (Fig. 2) consists of the following major parts:

- *Schematic View* panel provides the schematic representation of the target system and the graphical simulation data. The internal structure of the *data path* is also reflected there.
- *Microprogram table* is used to define the *control path* of the system. During the simulation this panel shows which line of the microprogram is currently executed.
- *Simulation* and *Test* tab-panels allow to choose and run RT-level fault and fault-free simulation. The simulation can be performed for a single set of input data (step-by-step or at once) as well as for all the sequence of input operands at once.
- *Simulation Results* tab-panel reflects the results of fault-free simulation.
- *Fault simulation* module provides fault simulation for the data path and its units.
- *Global Test Panel* is used to provide fault coverage information as for the whole data path as for each single unit under test.
- *Local Test Panel* provides means for manual local test patterns generation for a selected unit of data path. It also displays the gate-level schematic of the unit and the fault coverage for each unit as well as for the data path as whole.
- *Test Microporogram* is used to organize separate test access to each selected functional unit of the data path.
- *BIST module* provides the basis to experiment with embedded self-test facilities. The user can select the BIST mode and specify locations of signature analyzers within the data path and the LFSR (Linear Feedback Shift Register) architecture for pseudo-random TPG.



Design Cost: 87
Simulated Clock Cycles: 0
Clock Cycle: 5

A (0 .. 15): 4
B (0 .. 15): 3
C (0 .. 15): 0
D (0 .. 15): 0

Addr	Next	F1(in)	F2(in)	F4(in1)	F4(in2)	IN	F1	F2	F3	F4	F3(out)	F4(out)	OUT	Input	C1	C2	C3	C4
1	2					REG2								A	X	X	X	X
2	3					REG3								B	X	X	X	X
3	4	REG3		REG1	REG2		SHR0			ADD	REG3	REG1			X	X	0	X
3	3	REG3		REG2			SHR0			SHL0	REG3	REG2			X	X	1	X
3	END												REG1		X	X	X	X
4	3			REG2						SHL0		REG2			X	X	X	X

Fig. 2 Training system

The applet has a flexible design. The RT-level system model, shown in Fig 2 is not mandatory. Should any other model be used, it must be only specified in a form of text-files. Then it can be loaded just as simple as the original one.

The applet has a built-in extendable collection of examples implementing different algorithms. They help users to understand principles the system operation. For connecting the system to other applications as well as for providing users with a possibility to save the results of their work for further use applet has a data import/export capability. It also has a built-in multilingual support.

3. RT-LEVEL DESIGN-ORIENTED SUBSYSTEM

Each functional unit (FU) of the *data path* F1..F4 contains a number of microoperations (functions: unary and binary), which are labeled by corresponding control signals activating chosen function. There is an overlap between possible functions of F4 and of F1, F2 and F3 to allow a parallelization of a given algorithm. The user can select one or more microoperations for each unit of data path when implementing his own algorithm (like multiplication, division etc.). Each microoperation has a gate-level implementation, and the number of gates determines its cost and in the end the final HW cost of the system. The user can select, thus, a particular implementation of his algorithm (like *I* or *M-automata*, sequential or parallel *IM-automata*) meeting either the cost or timing requirement. The speed (the number of clock cycles) of the algorithm is measured by simulation. The simulation is supported by an RT-level model of the system as a whole and by gate-level models of each microoperation in each FU.

The *control path* is a microprogrammed controller [6], which implements Mealy FSM (Final State Machine). The controller consists of a microprogram table and an interpreter. The microprogram is developed by the user to realize a given algorithm based on the selected resources of the data path. The user fills in the rows of microprogram table, which contain information about the address of the current and the next microinstruction, MUX and DMUX configurations, Data IN values, selection of functions in FUs (F1 to F4) at each microinstruction, and status signal configuration.

In Fig. 2 an example of algorithm of multiplication of two operands A and B is presented. The result of multiplication is stored in REG1 and fed out to the data output.

The *RT-Level simulation* is carried out at the higher level by using corresponding to functional units Java subroutines which are activated according to condition values by the control signals in the order given in the microprogram table. The simulation data is stored in the Simulation Results subpanel. This data reflects the states of all the registers, outputs of all the functional blocks, data input and output of the device, current states at each clock cycle and condition signals. The simulation data can be used by the student as a debugging info as well as for the improving the efficiency: the speed or the cost of the system.

For more details on design-oriented part of our system, please visit the dedicated web page [4] and turn to our previous article [5] in which we made emphasis in RT-level design. In the present article, on the contrary, we will concentrate mostly on test-related topics.

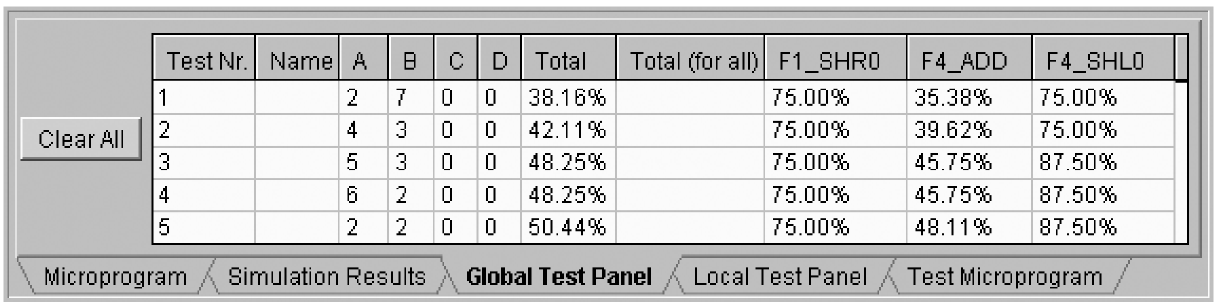
4. TEACHING RT-LEVEL TEST

The toolkit of the modern design and test engineer contains quite a few methods of testing of a SoC design. All of them have come from the earlier times and have been adopted for the new paradigm. With our teaching system we are aimed at showing a variety of different modern testing techniques including functional and deterministic testing, a number of BIST solutions.

Prior to entering the test mode, the system under test must be designed and verified. The user can do it himself or use one of prepared examples. When the test mode is selected, the microprogram and the structure of the data path are “frozen” and cannot be modified anymore. At the same time the user selects target microoperations of the data path for test generation and fault simulation. The fault simulation information is reflected (depending on a mode selected) at the Global Test Panel for the whole system and at the Local Test Panel for a single selected unit (Fig. 4). In the following we describe the test modes in detail.

Functional Test. In this mode the cheapest test technique is investigated, which does not require designing special test programs and embedding of special test structures into the system. The same unmodified microprogram and data path configuration are used instead. The required level of fault coverage must be achieved then by only a smart selection of input data. The sole checkpoint allowed for catching the fault is the data path primary output. Moreover, it only can be observed at the time when the microprogram outputs the final result.

The fault simulation information is presented at the Global Test Panel (Fig. 3). The input operands (A,B,C,D) are specified first. The same microprogram is used then repeatedly for fault simulation for all the input data. The fault coverage is calculated for each selected FU and for the whole system as well. The cumulative fault coverage for each input vector is



Test Nr.	Name	A	B	C	D	Total	Total (for all)	F1_SHR0	F4_ADD	F4_SHL0
1		2	7	0	0	38.16%		75.00%	35.38%	75.00%
2		4	3	0	0	42.11%		75.00%	39.62%	75.00%
3		5	3	0	0	48.25%		75.00%	45.75%	87.50%
4		6	2	0	0	48.25%		75.00%	45.75%	87.50%
5		2	2	0	0	50.44%		75.00%	48.11%	87.50%

Fig. 3 Global Fault Coverage table from the Global Test panel

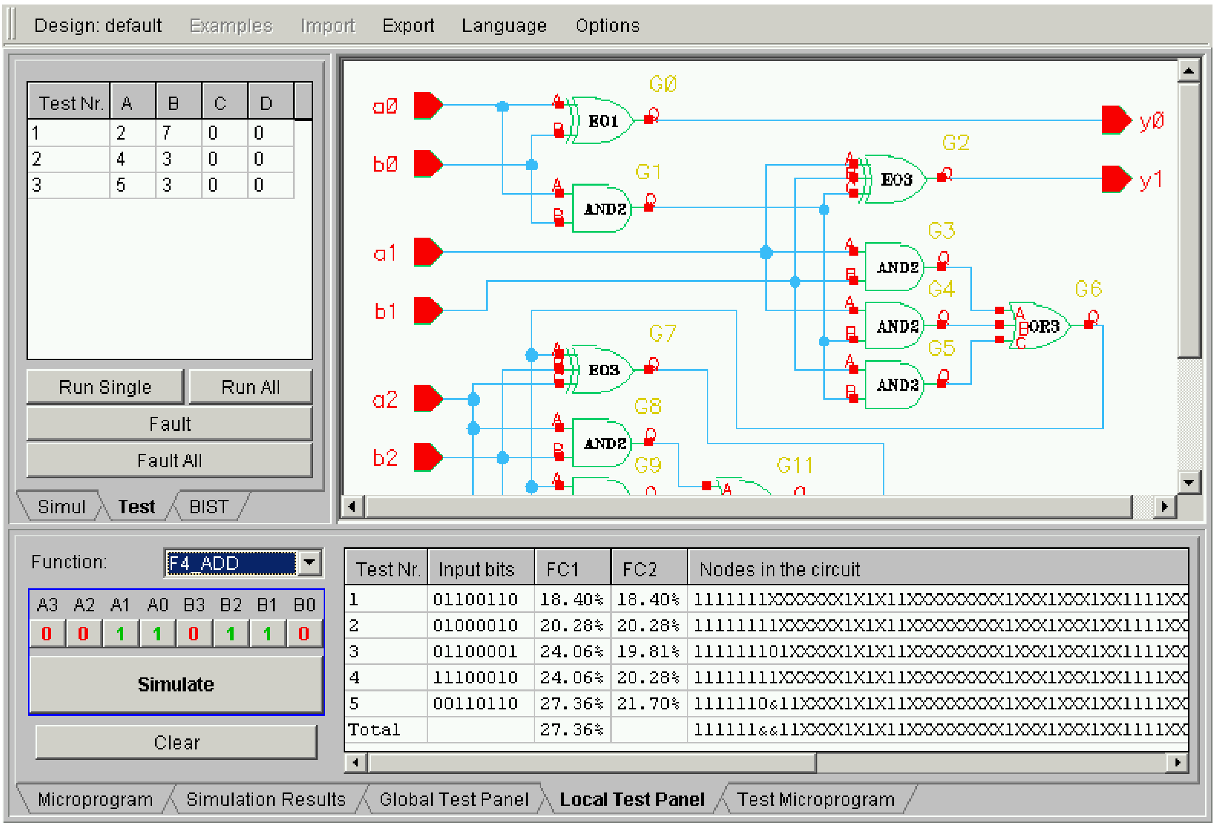
provided in the Global Fault Coverage table (Fig. 3).

The primary task of the student during investigation of functional testing is the selection of good operands in order to achieve the target fault coverage as fast as possible. For simpler designs this technique can be feasible. However, for more complicated structures something more sophisticated must be used.

Deterministic Test. This mode is aimed at a gate-level test generation and fault simulation for each selected FU separately. They are considered by the user in series and test vectors are generated. The simulation results are provided in the fault table at the Local Test Panel (Fig. 4). For each vector the fault coverage (FC2) is calculated and the information on tested nodes is given. The cumulative fault coverage (FC1) is also shown for each simulation step. The hierarchical RT-level fault simulation is also applied in order to evaluate the global fault coverage of those vectors for the data path as a whole. For this purposes a test program is composed for each selected FU. The simulation data is reflected in the Global Test Panel in the same way as it is done in the Functional Test mode.

In order to help the user generating gate-level test vectors, the gate-level schematic of currently selected FU is displayed. The user selects a target fault and generates a test vector. After pressing the "Simulate" button this vector is fault simulated at the gate level and the results (local fault coverage) are added into the fault table. At the same time, the same vector is sent to RT-level hierarchical fault simulator in order to fill in the Global Test Panel. In this panel the vector is shown as two decimal operands. The test microprogram, used for RT-level fault simulation must provide a good access to the selected FU. A simple version of such a program is generated automatically. It can be used as a template by a student in order to develop a more sophisticated test program if needed.

The primary task of the student working in Deterministic Test mode is the creation of short local tests covering maximum amount of faults for each of selected FUs. Another, more



Test Nr.	A	B	C	D
1	2	7	0	0
2	4	3	0	0
3	5	3	0	0

Function: F4_ADD

A3	A2	A1	A0	B3	B2	B1	B0
0	0	1	1	0	1	1	0

Test Nr.	Input bits	FC1	FC2	Nodes in the circuit
1	01100110	18.40%	18.40%	1111111XXXXXXX1X1X11XXXXXXXXX1XXX1XXX1XX1111XX
2	01000010	20.28%	20.28%	11111111XXXXXX1X1X11XXXXXXXXX1XXX1XXX1XX1111XX
3	01100001	24.06%	19.81%	111111101XXXXX1X1X11XXXXXXXXX1XXX1XXX1XX1111XX
4	11100010	24.06%	20.28%	11111111XXXXXX1X1X11XXXXXXXXX1XXX1XXX1XX1111XX
5	00110110	27.36%	21.70%	1111110&11XXXX1X1X11XXXXXXXXX1XXX1XXX1XX1111XX
Total		27.36%		111111&&11XXXX1X1X11XXXXXXXXX1XXX1XXX1XX1111XX

Fig. 4 Deterministic test pattern generation in Local Test Panel

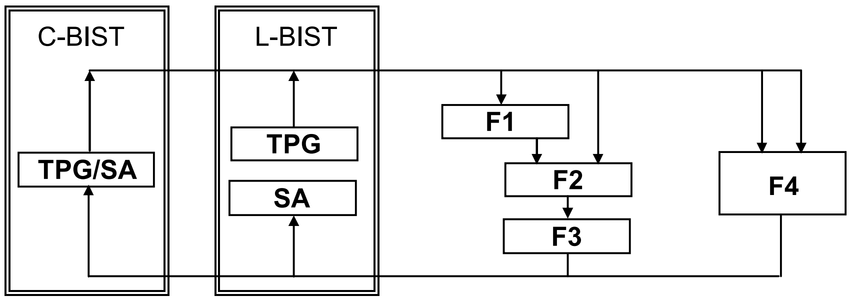


Fig. 5 Scan-path design

advanced, task is the reduction of the overall test length for the whole investigated system by modification of standard test programs and combination/modification of local test patterns. The latter allows testing faults in several FUs simultaneously and by the same test vectors.

The BIST Mode. The Deterministic Test mode is one of the most efficient ways of testing. However, it does not provide access to internal signals of the system under test. This problem is addressed by various BIST solutions. Usually it is a scan-path with a random test pattern generator (TPG) and one or more signature analyzers (SA). By scan-path technology (Fig. 5) the inputs and the outputs of the combinational blocks in data path are directly accessible by TPGs, SAs or TPG/SA (combined TPG and SA) [7].

Our teaching system allows reconfiguration of internal registers in the BIST mode. Depending on the chosen BIST method some of them can perform functions of TPG, SA or TPG/SA. If the *Logic BIST* (L-BIST) method is to be evaluated, the TPG and SA functions must be separated and located in different registers. On the contrary, in *Circular BIST* (C-BIST) both TPG and SA are situated in the same register. In the both modes it is possible to configure the TPG on-line from the interactive graphical panel. When the configuration is completed, the gate-level and the hierarchical fault simulation are performed and the results are displayed in the way similar to the one used in Functional and Deterministic test modes.

The described above modes help to illustrate the way of operation of different BIST structures and show how their efficiency depends on the TPG configuration. The selection of a good configuration for each selected FU is the main problem to solve by the student. Another task is the selection of such a single TPG configuration, that allows testing all of the FUs in the shortest possible time.

There is another BIST mode, called *Functional BIST* (F-BIST), implemented in the applet. This mode has very much in common to Functional Testing. The only difference between the two modes is that in the former one there is possibility to insert SAs at any arbitrary point within the data path. In this way we increase the observability of the system, since each such SA is capable of collecting data at each clock compressing it into an observable signature. The student's task then is to improve the efficiency of Functional Testing paradigm by introducing the minimal number of additional test points.

5. CONCLUSIONS

The conception presented allows to improve the skills of students in the area of digital hardware and SOC design connected with testing. The free-access basis and self-contained nature makes it easy for students even from foreign universities to use this system independently of time and place, and learn individually according to their own needs. This concept brings new forms of communication between teachers and students and up-to-date course material. The system's built-in multilingual support ensures easy integration into teaching courses of universities over the world.

The applet fully reflects the "easy action and reaction" conception which was taken as the major target for its creation. Each field, each functional unit, and other modules are clickable. Their functions can be changed or further adjusted. The reaction on each action is instantly reflected by highlighting of selected modules.

On the other hand the tasks chosen for training represent simultaneously real research problems. This provides students with dynamic environment to experiment with and to find interesting solutions for stated problems.

Acknowledgements: This work is partially supported by the Thuringian Ministry of Science, Research and Art (Germany), by EU V Framework project REASON, and by the Estonian Science Foundation (grants No 3658, 4876 and 4300).

REFERENCES:

[1] M.L. Bushnell. Increasing Test Coverage in a VLSI Design Course. International Test Conference, Atlantic City, NJ, USA, 1999, p. 1133.

[2] J. Harrington. VLSI Design 101 – the Test Module. International Test Conference, Atlantic City, NJ, USA, 1999, p. 1134.

[3] H.-D. Wuttke, et al. Internet Based Education - An experimental environment for educational purposes. *Proc. of IASTED*, May 6-8, 1999, Philadelphia, USA, pp. 50-54.

[4] Teaching system URL: http://www.pld.ttu.ee/dildis/automata/applets/9/

[5] S. Devadze, et al. Web-based training system for teaching basics of RT-level digital design, test, and design for test. *Proc. of MIXDES*, June 20-22, 2002, Wroclaw, Poland.

[6] Armstrong J. R., Gray F. G. Structured logic design with VHDL. *Prentice-Hall*, Englewood Cliffs, 1993, 482 p.

[7] Abramovici M., Breuer M.A., and Friedman A.D. Digital systems testing and testable design. *IEEE Press,* New York, 1999, 652 p.