Turbo Tester - Diagnostic Package for Research and Training

M.Aarna[1], E.Ivask[1], A.Jutman[1], E.Orasson[1], J.Raik[1], R.Ubar[1], V.Vislogubov[1], H.-D.Wuttke[2]

[1]Tallinn Technical University
Raja 15, 12618 Tallinn, Estonia
tt@pld.ttu.ee

[2]Technical University Ilmenau
Helmholtzplatz 1, 98693 Ilmenau, Germany
Dieter.Wuttke@tu-ilmenau.de

Abstract– *This paper describes a diagnostic software package called Turbo Tester. It contains a variety of tools related to the area of testing and diagnosis of integrated circuits. The range of tools includes test generators, logic and fault simulators, a test optimizer, a module for hazard analysis, built-in self-test simulators, design verification and design error diagnosis tools. The range of compatible diagnostic tools forms, via their interaction and complementary operation, a homogeneous research environment, which provides good possibilities for experimental research. Due to this fact, there are a number of scientific papers became possible. These papers have been presented at international conferences and published in reviewed journals. We give a couple of examples of such experiments in this paper. We also describe some laboratory work scenarios for students.*

1. Introduction

The increasing complexity of VLSI circuits and transition to Systems-on-Chip (SoC) or even Networks-on-Chip (NoC) paradigm has made test generation one of the most complicated and time-consuming problems in the domain of digital design. The more complex are getting electronics systems, the more important become problems of test and design for testability, as costs of verification and testing are getting the major component of design and manufacturing costs of a new product. This fact makes the research in the area of testing and diagnosis of integrated circuits (IC) a very important topic for both the industry and the academy.

Commercial CAD systems for VLSI design and test are both costly and do not provide a good variety of competing or complementary approaches to a given particular problem. They usually have a stiff workflow of standard integrated tools bound together and should be executed accordingly to a certain scenario. It is good for a designer but not for a researcher whose main goal is the search for new efficient solutions.

During the last decade, many different low-cost tools running on PCs have been developed to fill this gap. They usually include the major basic tools needed for IC design: schematics capture, layout editors, simulators, and place and route tools. However, low-cost systems for solving a large class of tasks from the dependability and diagnostics area: test synthesis and analysis, fault diagnosis, testability analysis, built-in self-test (BIST), especially for research and educational purposes, are still missing. For this reason, a diagnostic software Turbo Tester (TT) is being developed in Tallinn Technical University.

In this paper we briefly describe the main functionality of the Turbo Tester package and suggest possible areas of experimental research where TT can be used. Compared to previous paper [6] the reader will find new aspects of application of related tools as well as description of new functionality added to the package since that time.

Another possible application field of the TT package is the education. Entering the SoC era means that the test must become now an integral part of the VLSI and system design courses. The next generation of engineers involved with System-on-Chip (SoC) technology should be made aware of the importance of test, and trained in test technology to enable them to produce high quality and defect-free products. Therefore, we have developed a set of scenarios of laboratory works, which makes use of different aspects of the TT package. In this paper we give a short overview of these scenarios, while their full version is available in the Web [14].

The TT software consists of the following test related tools: test generation by different algorithms (deterministic, random and genetic), test program optimization, fault simulation for combinational and sequential circuits, testability analysis and fault diagnosis. TT can read the schematic entries of various contemporary VLSI CAD tools, e.g. Cadence, Synopsys, Mentor Graphics, Viewlogic, Compass, OrCAD, etc. which makes TT independent of the existing design environment. There are Turbo Tester versions available for MS Windows, Linux, and Solaris operating systems. The software is free of charge and it can be downloaded from the Web [13].

In the next section we give a short overview of the whole TT package and its main tools. It is followed by Section 3, which describes some research experiments made with TT. The overview of laboratory work scenarios is given in Section 4. Section 5 is dedicated for conclusions.

2. Overview of Turbo Tester Package

The main set of functional modules of the Turbo Tester diagnostic software package includes test generators, logic and fault simulators, a test optimizer, a module for hazard analysis, linear feedback shift register (LFSR) emulators for BIST, design verification and design error diagnosis tools (see Fig. 1).

The main advantage of the system lies in the fact that different methods and algorithms for various test problems are implemented and can be investigated as separately of each other as working together in different combinations. The latter provides a variety of different approaches to solution optimization for a particular problem.

Model Synthesis. The component library of Turbo Tester consists of Binary Decision Diagram (BDD) representations for the library components of the circuits to be processed. The library is open and can be updated for new components. The model generator creates a BDD-representation of the design from the netlist of the design, produced by e.g. schematic editor. The special kind of BBDs is used in Turbo Tester. They are called Structurally Synthesized BDDs (SSBDD) and provide a uniform approach to solving a wide scale of test design tasks, based on a uniform model and a restricted set of standard procedures. Unlike traditional BDDs, SSBDDs support test synthesis for gate-level structural faults. Moreover, the design can be represented either at the gate-level or at the macro-level. The latter one is a somewhat higher representation level, where the basic elements are macros consisting of several gates at once. On the macro-level, a BDD is to be created for each macro, where one-to-one correspondence between signal paths in the macro and nodes in the BDD will be established. For some tasks, such representation gives faster runtimes at the same accuracy [8]. A hierarchical DD model, which combines RT-level DDs and binary DDs is also possible. This allows migration of methods developed for logical level also to higher (behavioral and register-transfer) levels, where tools for hierarchical test generation and simulation have already been implemented [3].

Test Generation. For automatic test pattern generation (ATPG), random, deterministic and genetic test pattern generators (TPG) are implemented [4]. Mixed TPG strategies based on different methods can also be investigated. Tests can be generated for both, combinational and sequential circuits. Stuck-at faults and transition faults can be considered. The number of faults to be processed at the macro level will be less than the number of faults at the gate level (each macro-level fault represents, in general, a subset of gate-level faults). This causes the increase in productivity of test generation at the macro level compared to that of the gate-level. The best test generation efficiency for complex systems can be achieved by using the hierarchical DD representation [3].

Test Pattern Analysis. There are single-fault simulation, parallel fault simulation, and critical path tracing fault analysis methods implemented in the system. These competing approaches can be investigated and compared for circuits of different complexities and structures. As the result of using these tools, fault tables are calculated and test quality is evaluated for given test sequences. In a defect-oriented simulation mode the fault simulator uses a special defect library [1]. The physical defect model includes short (or bridging) faults and will be soon extended by open faults.

Test Set Optimization. The tool minimizes the number of test patterns in the test set by means of static compaction. The technique implements effective representation of fault matrices by weighted bipartite graphs. The approach contains a preprocessing step for determining the set of essential vectors. Subsequently, implications and a greedy search algorithm are applied. The proposed

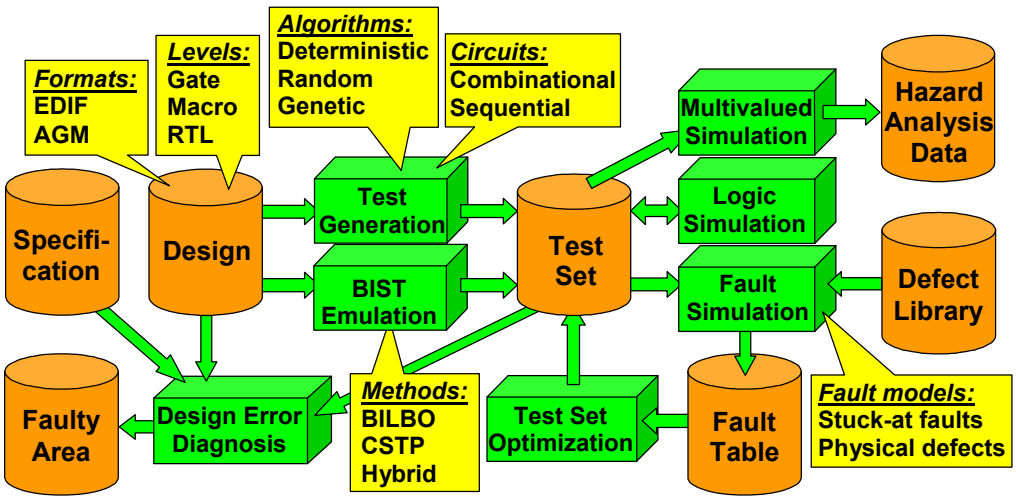

Fig. 1 Overview of Turbo Tester environment

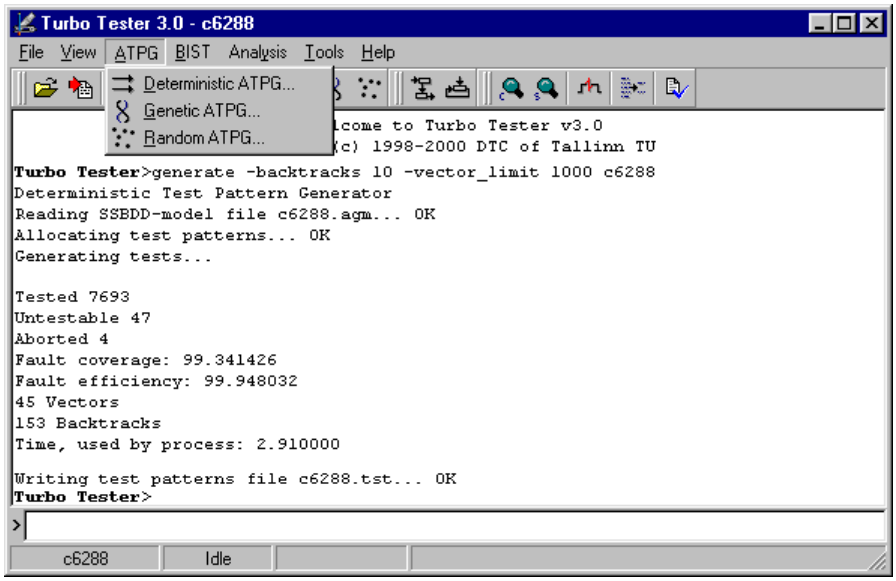

Fig. 2 Graphical user interface

method offers significantly fast performance in terms of run times [10].

Multi-valued Simulation. In Turbo Tester, multi-valued simulation is applied to model the possible hazards that can occur in logic circuits. The dynamic behavior of a logic network during one single transition period can be describes by a representative waveform on the output or simply by a corresponding logic value. In other words, each waveform type has a corresponding symbol of some given alphabet. Turbo Tester's multi-valued simulator implements 5-valued and 8-valued alphabets [11].

Design Error Diagnosis. After a digital system has been designed according to specifications, it might go through a refinement process in order to be consistent with certain design requirements (e.g. timing specifications). The changes introduced by this process (by a human or a CAD system) may lead to undesired functional inconsistencies compared to the original design. Such design errors should be identified via design verification. A design error diagnosis technique should be applied afterwards in order to locate and correct the error. In Turbo Tester we use the same SSBDD model for both the specification (the design before modifications) and the design to be corrected. An advantage of our particular approach is the fact that it does not need a special diagnostic test to be created. It uses a normal test set instead [9].

Testability Analysis. The real cost of a digital product is expressed as: *Cost(Design + Test)* < *Cost(Design) + Cost(Test)*. It follows from the fact, that the total product cost can be minimized by regarding the design and test of a product as one integral activity rather than the two disjoint unrelated activities. The latter approach is called design for testability (DFT). Among the most promising DFT methods are those aimed at enhancing the testability through adding redundant hardware elements or test-points (additional outputs for observing; inputs for controlling; additional flip-flops in scan-path etc.) to the circuit. The testability analysis tools of the system can be used for enumerating untestable faults, for selecting statistically hard-to-test faults, and for estimating the controllability, observability and testability characteristics for the nodes of the design. The tools are used for finding out where to alter the design to improve the testability.

Evaluation of Built-In Self Test (BIST) Quality. The BIST approach is represented by applications for Built-In Logic Block Observer (BILBO) and Circular Self-Test Path (CSTP) emulation. Different BIST architectures can be simulated and the self-test quality of these architectures can be evaluated. There is a tool, which utilizes a genetic search algorithm for automatically finding good BIST architectures. It is possible to use also the general "store-and-generate" approach, where the whole test sequence will be generated on the basis of a given set of test vectors (i.e. the stored part of the test). All these vectors serve as initial input test patterns for on-line test generation by BILBO or CSTP (i.e. the generated part of the test). A Hybrid BIST technique represents an opposite approach, which also partially utilizes deterministic patterns but in the very end of the sequence. This makes it possible to achieve higher fault coverage by shorter test sequence [7]. In Section 3 we discuss the Hybrid BIST framework in more detail.

Design Interface. Turbo Tester has a powerful design interface from EDIF 2.0.0 netlist format, which supports both, combinational and sequential designs. In this way, TT can read the schematic entries of various contemporary VLSI CAD tools, e.g. Cadence, Synopsys, Mentor Graphics,

Viewlogic, Compass, OrCAD etc., which makes the system open to different design environments.

Graphical User Interface. Turbo Tester Graphical User Interface (GUI) is under development. The current working version is shown in Figure 2. It is available for MS Windows OS only. Similarly to most of the contemporary CAD systems, TT has a dedicated shell window with a command prompt. The Turbo Tester tools can be executed as from this command prompt as by selecting corresponding entries from corresponding menus. The process output is displayed on the shell window. There is a handy visualization utility called Waveform Viewer (see Fig. 3), which illustrates properties of test sequences obtained with different test pattern generators.

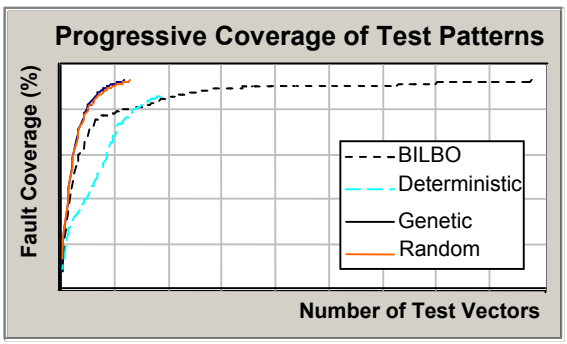

Fig.3 An example of test data representation in TT

WEB Interface. An Internet version of the Turbo Tester system (Web-TT) is available now as well. This new Web-based interface has the same functionality as the standalone TT. Users work with Web-TT by using simple HTML web pages via HTTP Internet protocol and process their data and results even without installing this system on their local PC. The user's OS and system requirements are not critical because TT performs all tasks on the remote server machine. Basically, an Internet connection and some Web browser (such as Netscape or MS Internet Explorer) at user's disposal are enough for the using of this system. An entry point of the Web-TT is a Welcome page (Fig. 4). This page contains login form, system overview and site navigation help. The module selection page contains a list of all available TT modules. User should select one of them and step through a set of initial parameters for the module. Every module has its own number and types of initial parameters that user has to set. Then the task will be submitted to the system for the execution. For the monitoring and administrating of user tasks there is a status page. User has an opportunity to observe the state of all his tasks there. After the task is performed user can download its results at any time from the status page. Web-based TT system provides an attractive alternative to the standalone version that offers separation of user interface from the low-level logic, easy of administrate and support, extensibility, and most importantly easy of use. Moreover, users will always work with the latest version of TT and does not need to download or install the system locally.

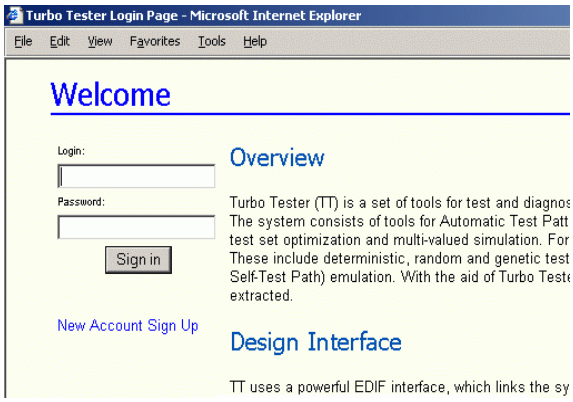

Fig. 4 Welcome page fragment of TT Web interface

System Portability. At present, Turbo Tester can be installed under MS Windows, Linux, and Solaris 2.x operating systems.

User Documentation. Turbo Tester installation includes a comprehensive reference manual [12], where all the functions of the system are explained. The manual is constantly updated together with the Turbo Tester package. It is designed in a style that is common to most of the CAD system documentations. The document complies partly with IEEE standard Std 1063-1987 for software user documentation. The manual is available at [13].

3. Research Experiments with TT

There is a number of scientific papers describing research carried out using Turbo Tester that have been published in international conferences as well as reviewed journals [1,3,4,5,7,8,9,10,11]. In this section, we give a couple of examples of possible research experiments with Turbo Tester.

Since test generation is one of the most important steps in the whole diagnostic framework, let us consider properties of the three described above combinational ATPGs available in TT package. Table 1 provides information about test quality and test generation time for ISCAS'85 benchmark circuits [2]. The test quality is represented by fault coverage (FC) and test length (TL). It should be mentioned that FC was measured on SSBDD model, where a compacted fault list is used [8]. Therefore, the percentage shows the ratio between detected representative faults and whole compacted fault set. The test time was measured on a PC with 800 MHz Pentium III processor, 256 MB RAM under MS Windows OS.

The deterministic ATPG was run in two modes: adjusted for a fast run and for a high FC. These modes are denoted in Table 1 by (1) and (2)

respectively. The simulation-based ATPGs (genetic and random) were adjusted to approximate the highest possible fault coverage with minimum amount of necessary test vectors. In most cases the genetic framework based ATPG was performing best by providing shortest test at the same or higher fault coverage. The best test generation time, however, was the one by the deterministic ATPG. In most cases it is several orders of magnitude shorter than that of other ATPGs. The reason, why the test length by that ATPG is not that good, lies in the fact that it is impossible to adjust it for test set minimization. Test compaction should be used after deterministic ATPG for that purposes.

	ATPG	Time, s	FC, %	TL
c432	Deterministic (1)	**0,03**	86,20	72
	Deterministic (2)	180,37	**93,02**	84
	Genetic	5,37	**93,02**	**34**
	Random	6,28	**93,02**	35
c499	Deterministic (1)	**0,07**	96,01	112
	Deterministic (2)	**0,07**	**99,33**	132
	Genetic	0,37	**99,33**	**84**
	Random	0,60	**99,33**	**84**
c880	Deterministic (1)	**0,02**	99,29	63
	Deterministic (2)	**0,02**	**100**	77
	Genetic	13,17	**100**	**33**
	Random	26,16	**100**	34
c1355	Deterministic (1)	**0,10**	93,20	93
	Deterministic (2)	**0,10**	**99,51**	124
	Genetic	0,56	**99,51**	**84**
	Random	0,65	**99,51**	**84**
c1908	Deterministic (1)	**0,08**	95,38	108
	Deterministic (2)	**0,08**	**99,48**	140
	Genetic	20,43	**99,48**	**106**
	Random	16,23	**99,48**	108
c2670	Deterministic (1)	**0,12**	93,60	134
	Deterministic (2)	0,21	**95,51**	155
	Genetic	42,22	95,39	**95**
	Random	373,73	95,39	104
c3540	Deterministic (1)	**0,17**	91,32	155
	Deterministic (2)	772,85	95,51	212
	Genetic	125,99	**95,54**	**108**
	Random	140,02	**95,54**	113
c5315	Deterministic (1)	**0,26**	92,83	103
	Deterministic (2)	9,87	**98,89**	171
	Genetic	140,91	**98,89**	**73**
	Random	200,60	**98,89**	79
c6288	Deterministic (1)	**0,17**	99,23	43
	Deterministic (2)	**0,17**	**99,34**	45
	Genetic	36,70	**99,34**	**16**
	Random	179,79	**99,34**	**16**
c7252	Deterministic (1)	**0,65**	92,76	168
	Deterministic (2)	837,31	**97,75**	279
	Genetic	262,07	96,66	**147**
	Random	581,74	96,48	183

Table 1 Experimental results by different ATPGs

Another good example of a research topic to be investigated using the Turbo Tester is the solution optimization in the Hybrid BIST framework. This approach makes use of cheap pseudorandom vectors at the first step when the fault coverage grows very fast and does not virtually depend on certain vectors. These vectors are generated on-line by an LFSR. At the second step, it applies a very limited amount of deterministic vectors that cover remaining hard-to-test faults. Such vectors are generated in advance and stored in memory.

There are several related issues which still have not found efficient solutions in the research and industrial community. One of such issues is the problem of finding the proper breakpoint between the first and the second parts of the test. This problem was illustrated in [7] and one of solutions was proposed there as well. The Turbo Tester's BIST emulator and the deterministic ATPG were used for this purpose. Figure 5 shows a graphical solution for this problem as the trade-off between the memory cost and testing time. Let have the whole cost of the BIST to be defined as

$$C_{TOTAL} = C_{TIME} + C_{HW} = \alpha T_G + \beta M_S$$

where C_{TIME} is the cost related to the time needed for test, C_{HW} is the hardware cost related to the BIST architecture, T_G is the length of the test generated by LFSR, M_S is the number of patterns to be stored, and α, β are constants to scale the test length and memory space. It would be very time consuming to find experimentally all the curves shown in Figure 5, except the generated test length T_G. The practical way is in trying to find the curve for M_S with as least as possible number of experiments, and to try to predict the curve on the basis of experimental data, and to approach then step by step to the real optimum by choosing as few as possible additional experiments. It is possible to solve such a task by Turbo Tester via writing a simple script, which has to analyze the results and to perform required steps.

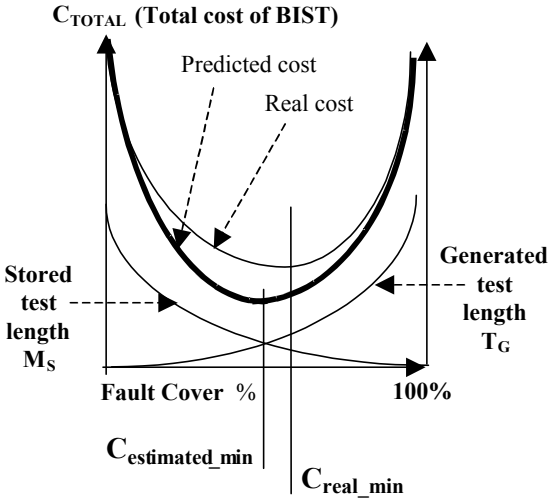

Fig. 5 Optimization of hybrid BIST

4. Laboratory Work Scenarios

The main aim of developed laboratory works scenarios is to teach and train students to integrate design and test, to give them knowledge on how to create testable designs or designs with self-testing capabilities, and how to obtain test patterns of better quality. The following laboratory works were

developed to train the engineering skills in the field of test:

- Test generation
- Design for testability
- Built-in self-test
- Design error diagnosis.

Test generation. The goal is to get acquainted with the problem and CAD tools of creating test patterns for digital circuits. At first, tests for the given circuit are generated manually. The fault simulation tool evaluates the quality of the manual tests. Then three different test-generating tools (based on deterministic, random and genetic algorithms) are used and compared with each other.

Design for testability. The goal is to show how the management of controllability and observability of test points in the circuit can improve the quality of testing. At first, a testability analysis is carried out for the given circuit by using test generation and fault simulation tools. Then, based on the testability information achieved, the circuit should be redesigned with the goal to get a test with a good quality i.e. with good fault coverage. Tradeoff problems between the redesign cost and test quality are investigated.

Built-in self-test. BIST is the capability of a circuit to test itself. Students concentrate themselves in an off-line BIST consisting of a test pattern generator (TPG), unit under test (UUT) and a response analyzer (RA).

TPG and RA usually are based on a LFSR. There are several disadvantages of such a structure: the tests generated usually are long, and they do not guarantee sufficient fault coverage. To overcome these drawbacks, a Hybrid BIST approach may be used. In this approach, a test engineer should solve the following problems:

- to find the best LSFR configuration for on-line test generation to achieve the highest fault coverage at the minimum length of pseudo-random test sequence;
- to find the best LFSR for response analysis to guarantee the minimum loss of accuracy in fault detection;
- to find the best level of mixing pseudo-random and stored tests as tradeoff between memory cost and testing time.

The task of the laboratory research for students is to find solutions for these problems. The students are not asked to carry out boring measurements, to simply press buttons for starting a program and getting results which are nothing but a simple confirmation of what they already know from lectures. Instead, they are asked to solve a series of engineering problems. They have a set of tools at their disposal and they have to plan and carry out experiments by themselves to find answers for the given questions.

Design error diagnosis. The goal is to learn how to compose diagnostic tests and to localize faults in a given circuit. Iterative using CAD tools, theoretical reasoning and manual work for generating additional "better" tests, students will get experience in solving extremely demanding engineering challenges.

The laboratory works have received good credits from students of Tallinn Technical University (Estonia), Darmstadt University of Technology (Germany), and Jonköping University (Sweden). It is under consideration to utilize TT for teaching Design for Testability in other universities of Eastern and Western Europe.

5. Conclusions

In this paper we have described a diagnostic software package called Turbo Tester, which has been developed in Tallinn Technical University. The package contains a variety of tools related to testing and diagnosis of VLSI circuits. The range of tools includes test generators, logic and fault simulators, a test optimizer, a module for hazard analysis, LFSR emulators for BIST, design verification and design error diagnosis tools.

The described extensive range of compatible diagnostic tools forms a homogeneous research environment via their interaction and complementary operation. Such a principle allows for interesting experimental research to be conducted. There are a number of scientific papers describing research carried out using the Turbo Tester environment. These papers have been presented at international conferences and published in reviewed journals.

Since the TT package provides a good environment for interesting laboratory work scenarios for students, we have developed such scenarios and we also described them briefly in current paper.

There are Turbo Tester versions available for MS Windows, Linux, and Solaris operating systems. The software can be downloaded from [13] free of charge.

Acknowledgements

This work was supported partly by the Thuringian Ministry of Science, Research and Art (Germany), by the EU Framework V project REASON, and by the Estonian Science Foundation Grant No 5649.

References

[1] M. Blyzniuk, FT. Cibakova, E. Gramatova, W. Kuzmicz, M. Lobur, W. Pleskacz, J. Raik, R. Ubar. "Hierarchical Defect-Oriented Fault Simulation for Digital Circuits," *IEEE*

European Test Workshop, Cascais, Portugal, Mai 23-26, 2000, pp.151-156.

[2] F. Brglez, H. Fujiwara, “A neutral netlist of 10 combinatori-al benchmark circuits and a target translator in FORTRAN,” *ISCAS, Special Session on ATPG and Fault Simulation*, 1985.

[3] M. Brik, G. Jervan, A. Markus, J. Raik, R. Ubar, "Hierarchical Test Generation for Digital Systems", *Mixed Design of Integrated Circuits*, pp. 131-136, Kluwer Academic Publishers, 1998.

[4] E. Ivask, J. Raik, R. Ubar. ”Comparison of Genetic and Random Techniques for Test Pattern Generation,” Proc. *of the 6th Baltic Electronics Conference,* Oct. 7-9, 1998, Tallinn, pp. 163-166.

[5] E. Ivask, J. Raik, R. Ubar. ”Fault Oriented Test Pattern Generation for Sequential Circuits Using Genetic Algorithms,” *IEEE European Test Workshop*, Cascais, Portugal, Mai 23-26, 2000, pp. 319-320.

[6] G. Jervan, A. Markus, P. Paomets, J. Raik, R. Ubar. “Turbo Tester: A CAD System for Teaching Digital Test,” *in "Microelectronics Education".* Kluwer Academic Publishers, pp.287-290, 1998.

[7] G. Jervan, Z. Peng, R. Ubar. ”Test Cost Minimization for Hybrid BIST,” *IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems.* Tokio, October 25-28, 2000, pp.283-291.

[8] A. Jutman, J. Raik, R. Ubar, "SSBDDs: Advantageous Model and Efficient Algorithms for Digital Circuit Modeling, Simulation & Test," in *Proc. of 5th International Workshop on Boolean Problems (IWSBP'02)*, Freiberg, Germany, Sept. 19-20, 2002, pp. 157-166.

[9] A. Jutman, R. Ubar, "Design Error Diagnosis in Digital Circuits with Stuck-at Fault Model," *Journal of Microelectronics Reliability*. Pergamon Press, Vol. 40, No 2, 2000, pp.307-320.

[10] A. Markus, J. Raik, R. Ubar. ”Fast and Efficient Static Compaction of Test Sequences Using Bipartite Graph Representation,” *Proc. of the Second Electronic Circuits and Systems Conference ECS'99*, pp. 17-20, Bratislava, Slovakia, Sept. 6-8, 1999.

[11] R. Ubar. “Dynamic Analysis of Digital Circuits with Multi-Valued Simulation,” *Microelectronics Journal,* Elsevier Science Ltd., Vol. 29, No. 11, Nov. 1998, pp.821-826.

[12] *Turbo Tester Reference Manual*, Version 02.10, Tallinn Technical University, Estonia, October 2002. Available at [13].

[13] Turbo Tester home page URL: http://www.pld.ttu.ee/tt

[14] Laboratory training URL: http://www.pld.ttu.ee/diagnostika/labs