Research Environment for Teaching Digital Test

E.Ivask, A.Jutman, E.Orasson, J.Raik, R. Ubar

Tallinn Technical University, Estonia
e-mail: raiub@pld.ttu.ee

D.Wuttke

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

Abstract. *A set of tools ("interactive modules") targeted to e-learning is presented for teaching test generation and fault diagnosis in digital circuits. The tools support university courses on digital electronics, testing and design for testability to learn by hands-on excercises test and fault diagnosis related topics. The tasks chosen for hands-on training represent simultaneously real research problems, which allow to foster in students critical thinking, problem solving skills and creativity.*

1. Introduction

The rapid developments in the areas of deep-submicron electron technology and design automation tools are enabling engineers to design larger and more complex integrated circuits. This progress is driving engineers towards design methodologies called System-on-Chip (SoC). The more complex are getting electronics systems the more important will be the problems of test and design for testability, as the expenses of verification and testing are becoming the major components of the design and manufacturing costs of new electronic products. The design and test cannot be seen any more as separate engineering issues. Entering into the SOC era means that test must become an integral part of the VLSI and system design courses.

At present, most system designers and electronics engineers know little about testing, so that companies frequently hire test experts to advise their designers on test problems, and they even pay a higher salary to the test experts than to their VLSI designers [1]. This reflects also today's university education: everyone learns about design, but only truly dedicated students learn test. 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 and defect-free products.

In this paper a conception is presented how to improve the skills of students to be educated for hardware and SOC design in test related topics. We first, present a learning method based on using so-called *living pictures* [2]. The method presented deals with the goal, to put interactive teaching modules to the Internet that can be used in a lecture as well as for individual self-studies. Second, we present a description of a laboratory research environment where the students can obtain hands on experimental and research experience in the field of design for test of digital systems [3].

2. State-of-the-art

In Fig.1 a general view on the traditional test flow and on the relationships between test related tasks is given.

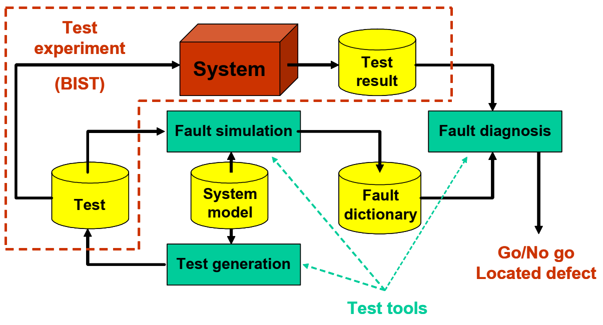


Fig.1. General view on the test flow and test tasks

For testing a system we need tests (test patterns, test sequences, test programs). For generating tests we need tools – test generators. To evaluate the quality of tests we need again tools – fault simulators. The efficiency and accuracy of test generators and fault simulators are depending on the models. There are a lot of possibilities how to model digital systems and the faults. Testing can be carried out either by external automated test equipments (ATE) or by built-in self-testing (BIST) facilities. There are many possibilities and ideas how to implement BIST. This is an emerging topic in the field of test, a way towards creating fault tolerant systems. To design BIST and evaluate its quality, again test generating and fault simulating tools are needed. To interprete the results of test sessions, fault diagnosis tools are needed.

To get experienced in such a complex field like testing, it is not enough to know how to work with tools in a manner of pushing on bottoms like usually commercial tools are exploited. To master the field, the student has to understand how the tools or algorithms are working, why and when some algorithms are more efficient than others. Commercial tools are not suitable to support interactive working sessions with the goal to understand the essence of algorithms inherent in tools. This was the motivation to create a research environment to support teaching digital design and test.

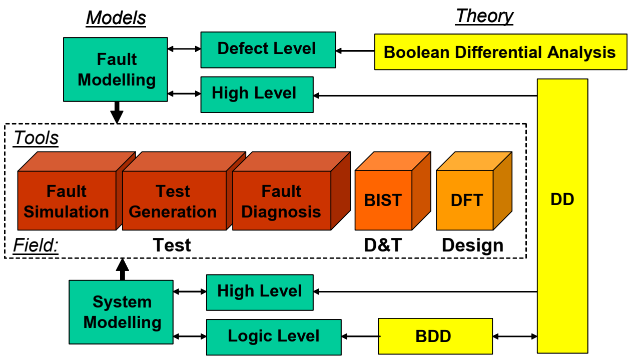


Fig.2. Topics map for teaching digital design and test

Main principles of test generation and fault diagnosis are the fundamentals of all the modern VLSI CAD and diagnostic software. Good understanding of these basics gives students vital background and skills in their future profession. Such basic aspects of testing should undoubtedly be taught to today's students as: manual and pseudo-random test pattern generation (TPG), fault simulation, combinational and sequential fault diagnosis, investigation of different BIST strategies.

The most important question in testing today's complex digital systems is: how to improve the testing quality at continuously increasing complexities of systems? Two main trends can be observed when searching solutions for the formulated problem: defect-orientation, and high-level modelling.

Traditional plain low-level test generation methods and tools for complex digital systems have lost their importance, other approaches based mainly on higher level functional and behavioral methods are gaining more and more popularity [4-5]. However, the trend towards higher level modelling moves us even more away from the real life of defects and, hence, from accuracy of testing. To handle adequately defects in deep-submicron technologies, new fault models and defect-oriented test methods should be used. But, the defect-orientation is increasing even more the complexity. To get out from the deadlock, these two opposite trends – high-level modelling and defect-orientation – should be combined into hierarchical approaches. The advantage of hierarchical approaches compared to the high-level functional modelling lies in the possibility of constructing test plans on higher functional levels, and modelling faults on more detailed lower levels.

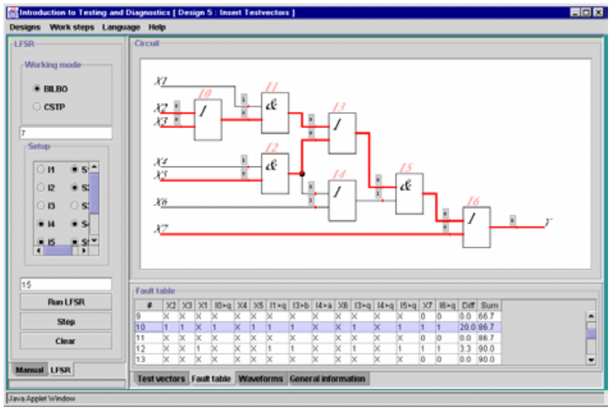
In Fig.2 a possible map of topics for teaching digital design and test is depicted. The main targets in the field of testing are to have good tools for solving the following tasks: fault simulation, test generation, fault diagnosis, design of BIST, and design for testability. Efficient and accurate tools are based on adequate models to represent the system and as well the faults. Efficiency (high performance) of tools can be reached at today's complexities of systems only by hierarchical approaches. So we need efficient modelling approaches for both, low (logic) and high-level representations of systems. Different mathematical approaches can be taken to support hierarchical modeling. In this environment two mathematical models are used: Boolean differential algebra for mapping physical defects onto the logic level, and decision diagram based graph theory for modeling defective systems in a uniform way both, at low (logic) and high (register transfer, procedural or instruction set) levels.

To support a digital design and test course based on the map in Fig.2 a research environment has been developed for hands-on training in solving research-intensive test related tasks.

3. "Living pictures" for learning test

The teaching software developed supports the action based training via internet. It offers a set of tools to inspect the objective to be learned, access to multiple learning modules, a big reservoir of examples and the possibility to generate new ones. It provides easy action and reaction (click and watch) by using "*living pictures*", the possibility of distance learning, and learning by doing [6]. The core of that concept are some Java-applets (the interactive modules) running over network, using standard browsers like Netscape and Internet Explorer with Java 1.2 runtime plug-in, or with Java 2 applet viewer [2,7].

The software can be used for teaching the basics of Digital Test and Testable Design as illustrative tool explaining the problems of fault modeling, fault simulation, test generation and fault diagnosis. Work window of this program (Fig.3) consists of three parts - vector insertion panel, view panel for design schematics, and view panel for test vectors, fault tables and waveforms. Vector insertion panel has two subpanels - one for manually inserting vectors and another one for automated pseudorandom test generation by different methods. The boxes at the lines on schematics are clickable for inserting proper signals during test generation. The following topics are supported by the software: fault simulation, test generation, testability analysis and built-in self-test. The key problems can be taught and learned using the software on different examples.



#	X2	X3	X1	I0->q	X4	X5	I1->q	I3->b	I4->a	X6	I3->q	I4->q	I5->q	X7	I6->q	Diff	Sum
9	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	66.7
10	1	1	X	1	X	1	1	1	X	X	1	X	1	1	1	20.0	86.7
11	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	86.7
12	X	X	1	X	X	X	1	1	X	X	1	X	1	1	1	3.3	90.0
13	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0.0	90.0

Fig.3. Work window of "living pictures" for logic test

Training for using of two methods for *test generation* is supported by the applet:

- direct test vector insertion on inputs (on the vector insertion panel), and
- test generation by path activation in the circuit (on the schematics panel).

There is a possibility to emulate a Linear Feedback Shift Register (LFSR) for interactive analyzing of BIST sessions based on using pseudorandom test patterns. By changing the settings on the vector insertion panel we can emulate different feedback structures of the chosen BIST architecture.

In the fault *simulation mode*, a fault table is generated and shown on the data panel for all the test vectors created by the given moment. By selecting a test vector on the data panel, all the detected faults will be highlighted by colours on the schematic panel.

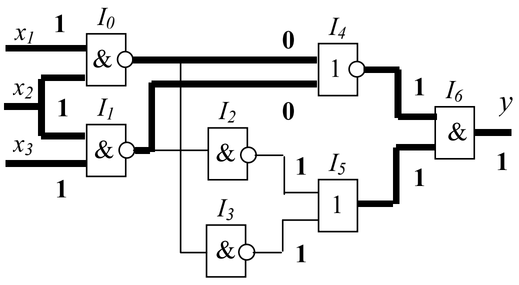


Fig.4. Fault simulation results

For example, in Fig. 4, activated paths (shown by bold lines) are found by fault simulating the test pattern $x_1x_2x_3$ = 111. The following faults are detected along these paths: $x_1 \equiv 0$, $x_2 \equiv 0$, $x_3 \equiv 0$, $I_{0b} \equiv 0$, $I_{1a} \equiv 0$, $I_{4a} \equiv 1$, $I_{4b} \equiv 1$, $I_{6a} \equiv 0$, $I_{6b} \equiv 0$, $y \equiv 0$. The inputs of gates are denoted from above down by *a,b*.

The following *fault diagnosis* strategies chosen from menu can be investigated: combinational or sequential one.

For learning the *combinational* diagnostic strategy, a single vector or a subset of vectors can be selected and applied to the erroneous circuit. The applet shows the results of testing, and displays also the subset of suspected faults. To improve the diagnostic resolution, additional test vector(s) may be generated and used in the repeated test experiment.

Sequential diagnosis (guided-probe testing) is based on the guided probing strategy. A test pattern is applied and the expected behavior of the circuit is displayed. The principle of guided-probe testing is to backtrace an error from the output where it has been observed to its source (faulty gate). By clicking on the connection boxes, the real values of signals of the faulty circuit can be measured. A faulty gate is located if it has been found that the signal on the output of the gate is faulty, while only expected signals are observed at its inputs.

The main didactive point in learning the both diagnostic strategies is to try to localize the fault by as few test vectors (in the combinational approach) or by as few measurements (in the case of sequential approach) as possible. In this task a competition between students can be carried out which makes the "play" with the applet even more exciting.

4. Tool environment for training research

Traditional VLSI test generation and fault simulation software on workstations are both costly and unable to handle large numbers of students simultaneously in educational courses. During the recent years, many different low-cost tools running on PCs have been developed to fill this gap. They include usually the major basic tools needed for IC design: schematic capture, layout editors, simulators and place and route tools. Low-cost systems for solving a large class of tasks from the testing area - test generation, fault simulation, fault diagnosis, testability analysis, built-in self-test, especially for teaching purposes, are missing. For this reasons, a tool environment has been developed (Fig.5) for training research in the test field [3,8].

After learning theoretical topics described in the previous section, a laboratory research follows with more complex and realistic designs.

The main originality and added value of the environment are in different optional methods implemented to give students opportunity to compare different approaches and algorithms used in the testing practice. The students develop their own circuits as diagnostic objectives, investigate the testability of circuits, redesign them if necessary for improving the testability, insert self-test BIST structures, analyze the efficiencies and trade-offs of different BIST solutions and learn to make proper engineering decisions in the field of testable design. The digital system developed can be represented at different levels (logic or macro levels) to give a possibility to investigate how the performance of tools is depending on the complexity of the model.

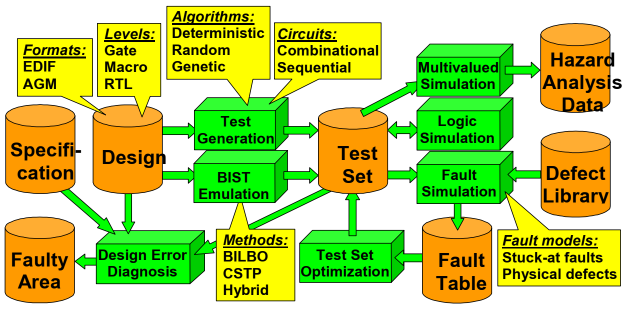


Fig.5. Overview of the tool environment

The main set of functional modules of the environment 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 (Fig. 5).

System modeling. To implement such a large set of tools in a university environment was possible because of using a single library of components for implementing different algorithms. Traditionally for solving the mentioned set of tasks the following library types are used: Boolean expressions for simulating, D-cubes for test generating, truth tables in multi-valued logic for multi-valued simulation and hazard analysis, dedicated set-theoretical expressions for deductive fault analysis etc. In the case of the given environment, all the tools are using a single library based on Binary Decision Diagrams (BDD). Using BDDs makes it possible to create models with different complexities (networks of macros with different complexities) whereas there is no need to introduce new models for macros into the library (the models of macros are always synthesized automatically, and each macro will be represented by Structurally Synthesized BDDS (SSBDD) [10]). Unlike traditional BDDs, SSBDDs support structural modeling of macros for test generation, fault simulation, delay fault analysis, or hazard checking purposes. Moreover, the design can be represented either on the gate-level or on the macro-level to reduce the complexity of the model compared with gate-level modeling. For some tasks, such representation gives faster runtimes at the same accuracy [11]. 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 [12,13].

Test Generation. For automatic test pattern generation (ATPG), random, deterministic and genetic test pattern generators (TPG) are implemented [14]. 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 [12].

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 [15]. The physical defect model includes short (or bridging) faults and will be soon extended by open faults.

Tools for Built-in Self-Test analysis. Because of the high cost of external testers and of inefficiency of testing in real speed by external testers BIST is the emerging testing concept in VLSI testing and, particularly, in the System on Chip field. BIST is the capability of a circuit to test itself. BIST is usually based either on generating on-line pseudorandom patterns by so called Linear-Feedback-Shift-Registers (LFSR) or on using functional test patterns. In both cases it is difficult to reach high-fault coverage because of existing hard-to-detect faults which can be detected only by a single or very few test patterns. To reach high fault coverage, the HW-based generated pseudorandom (or functional) test patterns can be complemented by deterministically pregenerated and stored test patterns. This combination is called – hybrid BIST. The problem is here to find out the best and cost-effective combination of on-line and stored test patterns.

Tools for design error analysis. Design errors stand for the mistakes or faults introduced by a designer or a CAD system during the design process. Such errors usually manifest themselves during design validation and verification stage. If the specification and the implementation of a design do not match a design error has been introduced. An example of a design error could be a gate substitution error. For example, an AND gate is replaced by OR gate in the implementation. It could be also a bad or missing connection between gates. Extra or missing inverters are also examples of a design error.

Design error diagnosis is an illustrative area for teaching diagnosis. It shows both the essential principles of fault diagnosis and the existence of alternative fault models other than stuck-at faults. The latter is important because the stuck-at fault model is widely accepted and taught over the world. Therefore students know usually very little about the existence of different fault models and different diagnostic problems.

A detailed description of other tools in the environment can be found in [8,9]. A description of different research scenarios created for the environment is given in [16].

5. Conclusions

An environment and a set of tools are presented for improving the skills of students to be educated for hardware and SOC design in test related topics. Based on the described environment, an e-learning conception using simple “living pictures” on one hand, and hands-on training sessions on the other hand can be introduced into study programs at technical universities for teaching courses on digital electronics, testing and design for testability. The tasks chosen for hands-on training represent simultaneously real research problems, which allows to foster in students critical thinking, problem solving skills and creativity in a real research environment and atmosphere. 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.

Acknowledgements. The research has been partly funded by European projects REASON (IST-2000-30193), Evikings II (IST-2001-37592) and ESF grants G5649 and 5910.

References

1. M.L. Bushnell. Increasing Test Coverage in a VLSI Design Course. International Test Conference, Atlantic City, NJ, USA, 1999, p. 1133.
2. R.Ubar, H.-D.Wuttke. The DILDIS-Project – Using Applets for More Demonstrative Lectures in Digital Systems Design and Test. 31st ASEE/IEEE Frontiers in Education Conference. Abstracts, Oct. 10-13, 2001, Reno, NV, USA, pp.83.
3. R.Ubar, A.Jutman, E.Orasson, J.Raik, T.Evartson, H.-D.Wuttke. Internet-Based Software for Teaching Test of Digital Circuits. In the book "Microelectronics Education", Marcombo Boixareu Ed., 2002, pp.317-320.
4. M.L.Bushnell, V.D.Agrawal. Essentials of Electronic testing. Kluwer Acad. Publishers, 2000, 690 p.
5. S.Mourad, Y.Zorian. Principles of Testing Electronic Systems. J.Wiley & Sons, Inc. New York, 2000, 420 p.
6. H.-D. Wuttke, K. Henke, R. Peukert. Internet Based Education - An Experimental Environment for Various Educational Purposes. Proc. of the IASTED Int. Conf. on Computers and Advanced Technology in Education, May 6-8, Philadelphia, PA USA, 1999.
7. R.Ubar, E.Orasson. E-Learning tool and Exercises for Teaching Digital Test. Proc.of 2nd IEEE Conf. on Signals, Systems, Decision and Information Technology. Sousse, Tunisia, March 26-28, 2003, CIT-6, pp.1-6.
8. Turbo Tester Reference Manual, Version 02.10, Tallinn Technical University, Estonia, December 2003. Available at [13].
9. Turbo Tester home page URL: http://www.pld.ttu.ee/tt
10. 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.
11. 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.
12. J.Raik, R.Ubar. Fast Test Pattern Generation for Sequential Circuits Using Decision Diagram Representations. Journal of Electronic Testing: Theory and Applications. Kluwer Academic Publishers. Vol. 16, No. 3, pp. 213-226, 2000.
13. R.Ubar, A.Morawiec, J.Raik. Cycle-Based Simulation Algorithms for Digital Systems Using High-Level Decision Diagrams. IEEE Proc. of Design Automation and Test in Europe. Paris, March 27-30, 2000, pp. 743.
14. 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.
15. 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.
16. M.Aarna, E.Ivask, A.Jutman, E.Orasson, J.Raik, R.Ubar, V.Vislogubov, H.D.Wuttke. Turbo Tester – Diagnostic Package for Research and Training. J. of Radioelectronics and Informatics, No3 (24), July – September, 2003, pp. 69-73.