E-Learning tool and Excercises for Teaching Digital Test

Raimund Ubar, Elmet Orasson
Tallinn Technical University
Raja 15, 12618 Tallinn, Estonia {raiub,elmet}@pld.ttu.ee

Abstract. *A set of tools ("interactive modules") targeted to e-learning is presented for teaching logic level test generation and fault diagnosis in digital circuits. The tools support different university courses on computer engineering, switching and automata theories, digital electronics and design for testability to learn by hands-on excercises test and fault diagnosis related topics. A big reservoir of examples and the possibility to set up interesting engineering problems like how to generate test patterns for a digital circuit, or how to locate a faulty gate makes the learning process more interesting and allows learning at an individual depth and duration. The interactive modules are focused on easy action and reaction, multilingual descriptions, learning by doing, and a game-like use. 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 increasing complexity of VLSI circuits, Systems-on-Chip (SOC) or even Networks-on-Chip (NOC) has made test generation one of the most complicated and time-consuming problems in digital design. The more complex are getting electronics systems, the more important will be the problems of test and design for testability because of the very high cost of testing electronic products. 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 System-on-Chip (SoC) technology should be made better 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 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 [2]. They can be accessed independent of time and place. On one hand, teachers can demonstrate different examples and procedures of test related topics using living pictures during the lessons. On the other hand, students can use the same simulations on their home computer, if the living pictures are available on the Internet.

The core of the teaching concept presented are some JAVA-applets (the interactive modules) running on any browser connected to the Internet. We call this type of applet "Living Pictures". By using interaction possibilities the students can produce input stimuli, watch the behaviour of the circuit in the fault-free mode and also in different faulty modes. In the paper, different learning tasks and excercises are described which make use of this applet.

2. User interface

The program for representing "living pictures" for teaching Digital Test is written in Java 1.3. It can be run over network, using standard browsers like Netscape and Internet Explorer with Java 1.3 runtime plug-in, or with Java 1.3 applet viewer. The program can be used for teaching the basics of testing digital systems, deterministic test generation, pseudo-random test generation, fault simulation and fault diagnosis.

The work window of the applet consists of three main parts (Fig.1):

- Vector insertion panel
- View panel for design schematics
- View panel for displaying information (test patterns, fault tables, waveforms, and different statistics)

The *vector insertion panel* has two subpanels for inserting single input test vectors and for setting up the feedback configuration of a Linear Feedback Shift Register (LFSR) to be used for automatically generating test vectors [3]. In the LFSR mode, the first subpanel is used for initializing the LFSR. The LFSR based Automated Test Pattern Generator (ATPG) is used for emulating different Built-In Self-Test (BIST) ideas like BILBO, Circular-Self-Test-Path (CSTP), "Store-and-generate" [3], or other hybrid BIST approaches [4].

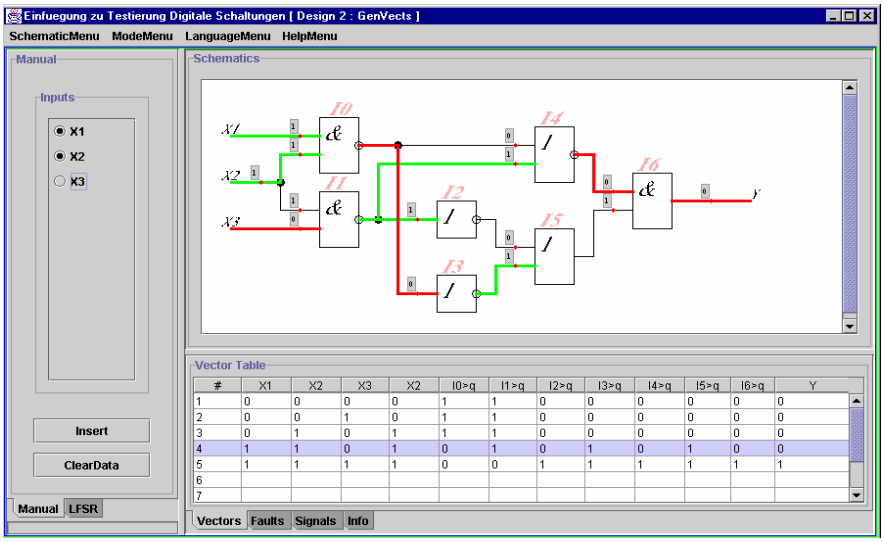

Fig.1. Working windows of the applet

The first subpanel is also used when creating test vectors for specific fault detection. In this case, the fault activating and propagating values are inserted one by one into the signal boxes at connections of the design schematics, and the input test vector will be deduced from these internal signal values.

The *schematics panel* displays currently selected schematics. The small boxes at the lines display internal signal values on connections. The boxes are clickable during manual test vector generation and fault diagnosis. In the test generation mode, the needed signal values for fault activation or fault propagation can be inserted directly at the connections. In the fault diagnosis mode, by clicking the boxes, a guided probing procedure can be simulated. A click on the box shows the result of measuring the "real" signal on the corresponding connection of the simulated faulty circuit.

Detected faults, signal conflicts etc. are displayed as colored bold wires. Color coding is as following:

- red - stuck-at-1 fault is detectable,
- green - stuck-at-0 fault is detectable,
- gray - undefined (don't care) signal, and
- blue - conflicting signals.

The *data panel* displays information about simulated test vectors and detected faults. In the fault simulation mode you can click on the row of a given test vector and have a visualization which faults are detected by the given vector. In the signal (waveform) mode you can select all the signals in interest and leave out those which are not.

There are four main menus used with the applet: schematics, mode, language, and help.

The *schematics menu* contains a list of predefined circuits. For didactive purposes most of them are very simple circuits for better understanding the most impüortant relationships between signals, functions and faults.

By the *language menu* the user may choose one of the currently supported languages from the given list.

The *help menu* provides with useful tips and explanations.

The *mode menu* tells the applet what is to be done

- test vector insertion,
- manual test vector generation,
- fault simulation or fault diagnosis (two possible diagnostic approaches are implemented: sequential and combinational diagnosis).

We start working with the applet by selecting a circuit from a set of predefined ones. Then we can carry out different experiments with this circuit by selecting a proper working mode from the mode menu.

3. Test vector generation

There are two methods possible for test vector generation using the applet:

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

In the *direct test vector insertion mode* we can choose test vectors either automatically by using LFSR, or by inserting vectors manually.

In the manual mode, we generate step by step input patterns which are simultaneously simulated. The boxes at the lines on the schematics subpanel display the result of simulation – the values of internal signals on the connections. The waveforms can be viewed on the data subpanel.

When using LFSR, we have to specify the initial state, to set up the feedback structure, and to specify the length of the test sequence. By LFSR we can simulate the BIST either in the mode of Built-In Logic Block Observer (BILBO) or in the mode of Circular Self-Test Path (CSTP) [4,5]. By changing the settings on the vector insertion panel we can emulate different feedback structures of the chosen BIST architecture.

In the *test generation mode* we choose a target fault in the schematic and create step by step proper activated paths in the ciruit to activate the fault at his site and to propagate the error signals caused by the fault towards output by clicking the needed values into boxes on the lines. From these values finally, an input vector will be deduced. The colours on lines help us to understand the current status of the task: activated faults and activated paths are marked by red and green lines, the inconsistencies of the signal values are highlighted by blue colour. As the result of the procedure, a test pattern will be generated. The detected by the test faults are displayed also on the data panel in form of a row in the fault table.

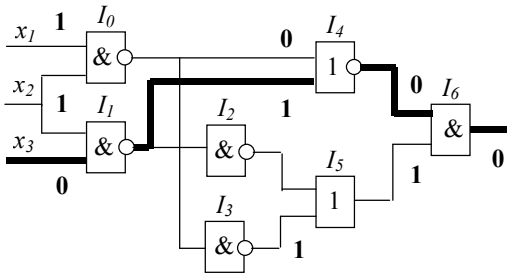

Fig.2. Test generation by path activation

For example, to generate a test pattern for the fault $x_3 \equiv 1$ in Fig. 2, first, a signal with opposite value 0 to the faulty value 1 should be inserted to x_3 by clicking the box on the line x_3. Then, the faulty signal of x_3 should be propagated to the output of the circuit. By inserting the value 1 on the line x_2 the faulty signal from x_3 is propagated through the gate I_1. Next, by inserting the value 0 on the upper input of gate I_4 the faulty signal is propagated through the gate I_4. Finally, by inserting the value 1 on the lower input of gate I_6 the faulty signal is propagated through the gate I_6 to the output y. The activated path is shown in Fig.2 by bold lines. All the inserted values should be properly justified step by step by other signals moving towards the inputs. As the result, a test pattern will be created on the inputs. For this example, the input pattern $x_1x_2x_3 = 110$ will be found.

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.

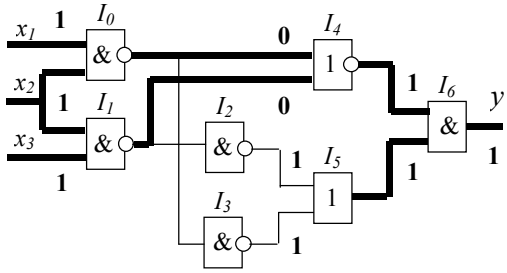

Fig.3. Fault simulation results

For example, in Fig. 3, 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*.

In Fig. 1, the results of fault simulation for 5 test vectors are shown. On the schematic panel we see the activated paths and detected faults for the vector number 4 which is selected in the view panel. The values in boxes show the behaviour of connection lines of the circuit for this test vector. The activated faults are highlighted by coloured lines, the value 0 (or 1) in the boxes means that the fault stuck-at-1 (or stuck-at-0) is activated.

4. Fault diagnosis

In the *fault diagnosis mode* we need at first, to create a fault table by running the fault simulator for a set of previously generated test vectors. Entering into the diagnosis mode will insert a random fault into the circuit.

The following diagnosis strategies chosen from menu can be investigated: combinational and sequential diagnosis.

For learning the *combinational* diagnostic strategy, a single vector or a subset of vectors can be

selected and applied to the erroneous circuit (by imitating test experiments). 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.

As an example, let us see the procedure of sequential fault localization by pinpointing the signals in the circuit for the case of test pattern $x_1x_2x_3 = 110$ represented in Fig.2. Suppose the gate I_1 is faulty. An error has been observed on the output. Three possible fault location procedures can be imagined.

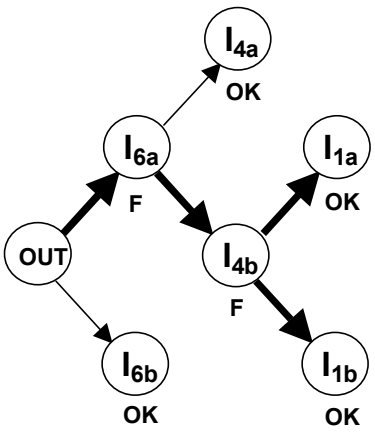

Fig.4. Fault diagnosis by backtracing errors

First, we may use a trivial backtracing procedure of erroneous signals shown as a search tree in Fig. 4. In the worst case we may click on the 6 nodes. Starting with the node I_{6b} we observe a correct signal. Then, we try the next input I_{6a} of the gate I_6 where the error is detected. We continue now backtracing in the node $I_{4\,a}$. Then, we try the next input I_{4b} of the gate I_4 where again the error is detected. Now we backtrace to the inputs of the gate I_1 where no errors are found. This means that we have located the erroneous gate I_1 by 6 measurements.

Second, we may analyse the fault activation conditions on the inputs of gates in the backtrace tree for local optimization (at each gate) of the search process. For example, based on the input signals of the gate I_6 we realize that if an erroneous signal has been propagated through the gate, it can originate only from the input I_{6a}. In other words, we can skip pinpointing of the node I_{6b}. In the same way, we realize that the measurement of I_{4a} is also not needed. As the result, the backtracing procedure will cost only 4 measurements (see the bold lines in Fig.4).

There is a third possibility to analyze the situation for a global optimization of the searh process. Since there exists a continuous activated path from the input x_3 to the output y, the faulty gate should be located on that path. By measuring the value of I_{4b} we can divide all the possible faults into two equal groups. In the case of correct value, we have to proceed towards the output and pinpoint the value of I_{6a} to determine which of the gates I_4 or I_6 is faulty. In the case of erroneous signal, we have to continue towards the inputs and measure the value of x_3 to determine if either the input x_3 or the gate I_1 is faulty. In both cases we need only 2 measurements to locate the fault.

On this little example, we managed to show that the fault diagnosis process can be regarded as a demanding mental experiment. A competition can be organized between students to make the learning procedure an exciting event.

5. Research training in BIST

In the following we show how the tasks can be chosen based on the applet for hands-on training, which simultaneously represent real research problems. Solving the formulated tasks allow to foster in students critical thinking, problem solving skills and creativity.

The research oriented tasks are related to the field of Built-In Self-Test (BIST) in Systems on Chip.

BIST is the capability of a circuit to test itself. From a large variety of BIST methodologies, we concentrate ourselves in a off-line BIST [5] consisting of the following main components: test pattern generator (TPG), unit under test (UUT) and a response analyser (RA). The corresponding BIST architecture is shown in Fig. 5.

TPG is usually a pseudorandom test pattern generator, and RA – a signature analyser, both based on linear feedback shift registers (LFSR) [5].

There are several disadvantages of such a structure. First, the test sequences generated randomly are usually very long, second, they do not guarantee always a sufficient fault coverage because of existence of so called "hard-to-test" faults.

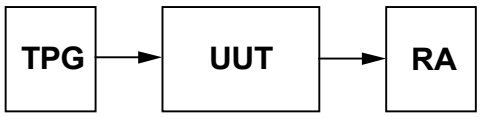

Fig. 5. BIST architecture

To overcome these drawbacks, combinations of several approaches have been proposed. One of them, called hybrid BIST, is based on combining on-line generated pseudorandom test patterns with stored pregenerated test patterns.

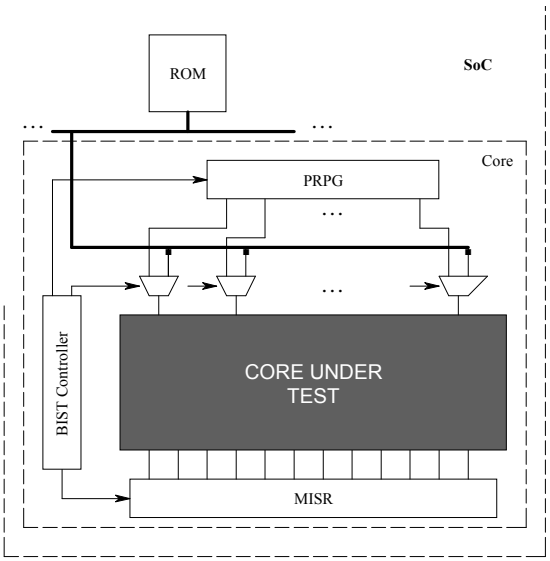

Fig. 6. BIST architecture

A hybrid BIST architecture is depicted in Fig.6. Pseudorandom pattern generator (PRPG) and Multiple Input Signature Analyzer (MISR) are implemented inside the core under test. Pre-generated deterministic patterns are stored in ROM.

In this approach, at first pseudorandom test sequence with a length L is generated on-line, after that a switch to a stored test approach takes place. For the stored test approach, previously generated and then in the memory stored test patterns are read one by one from the memory and applied to the UUT to reach the 100% fault coverage.

The applet presented allows to generate test patterns by both methods: by generating on-line pseudorandom test patterns using the LFSR approach, and by manually generating deterministic test patterns for the faults not detected by pseudorandom test patterns.

There are now several problems to be solved which still have not found sufficient solutions in the research and industrial community:

- What is the shortest LFSR and what is the best characteristic polynomial for the LFSR to be used for on-line test generation to achieve the highest fault coverage at the minimum length of the pseudorandom test sequence?
- How to find the best level of mixing the pseudorandom test and stored deterministic test as the tradeoff between the memory cost and testing time.

To find solutions for the mentioned questions will be the task of the laboratory research for students. The students are not asked to carry out boring measurements, to press simply on 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 to plan and carry out experiments to find answers for the given questions.

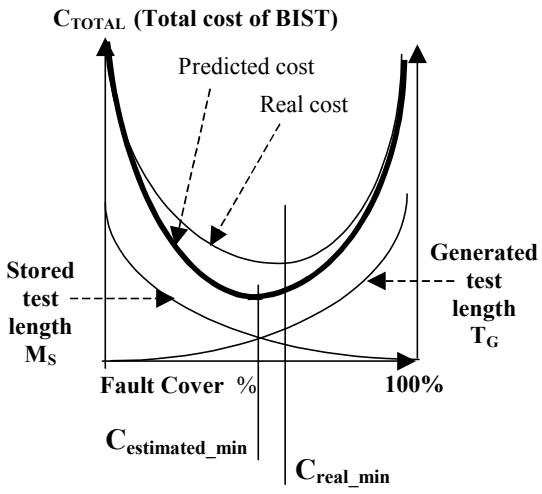

Fig. 7. Optimization of hybrid BIST

The are not available straightforward algorithms or software tools to find directly solutions for the

mentioned problems. The only method is to set up hypotheses and check them by experiments.

Fig. 7 shows a graphical solution for finding the optimum of mixing pseudorandom and stored test approaches as the tradeoff between the memory cost and testing time. Let have the whole cost of the BIST to be found 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 is the number of patterns to be stored, and α,β are constants to map the test length and memory space to the costs of two parts of test solutions to be mixed.

The problem is that it would be very time consuming to find experimentally all the curves shown in Fig. 7 except the generated test length T_G. The practical way would be 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 then by choosing as few as possible additional experiments to approach step by step to the real optimum.

To find a proper algorithm for solving this optimization problem will be the task of the student.

6. Conclusions

The described applet can be used for teaching the basics of testing digital systems.

The teacher can use the applet during the lecture explaining the basics of the topic. The applet can be used also during the exam for giving some tasks to students.

Students can use the same applet for training purposes. They can insert different possible faults, and watch how the faults change the circuit's behavior at different input patterns, how the test patterns can be generated to detect a given fault, or how the faults can be localized by test patterns.

The tasks formulated for students based on the applet are research oriented.. The students are not asked to carry out boring measurements, to press simply on 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 problems, and they themselves have to plan and carry out experiments to find answers for the given questions.

By the use of web-based media we achieve:

- presentation of course material independent of place and time,
- individual learning according to the students' own needs,
- new forms of communication between teachers and students (chat, joint editing),
- up-to-date course material.

The conception presented allows to improve the skills of students to be educated for digital hardware and SOC design in test related topics.

The principal mission of the conception is to inspire students to learn, to inspire them on a journey to knowledge, and to prepare them to develop problem-solving strategies.

Acknowledgement. This work was supported partly by the Thuringien Ministry of Science, Research and Art (Germany), by the EU Framework V project REASON, and by the Estonian Science Foundation Grant No 4300. We thank also Dr. Wuttke from TU Ilmenau, Germany for close cooperation in developing this applet.

References

1. M.L. Bushnell. Increasing Test Coverage in a VLSI Design Course. International Test Conference, Atlantic City, NJ, USA, 1999, p. 1133.
2. Ubar, R., Wuttke, H.-D., "Action Based Learning System For Teaching Digital Electronics And Test" ,*3rd European Workshop on Microelectronics Education" (EWME 2000), Aix-en-Provence, Kluwer Academic Publishers,* May 18-19 2000, pp. 107-110.
3. M.Abramovici et al. Digital Systems Testing and Testable Design. IEEE Press, 1999, 652 p.
4. G.Jervan, H.Kruus, Z.Peng, R.Ubar. About Cost Optimization of Hybrid BIST in Digital Systems. 3rd Int. Symp. on Quality of Electronic Design, San Jose, California, March 18-20, 2002, pp.273-279.
5. M.L. Bushnell, V.D. Agrawal. Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits. Kluwer Academic Publishers, 2000, 690 pp.
6. http://www.pld.ttu.ee:81/java/IntroI/Training.htm