THESIS ON INFORMATION AND SYSTEM ENGINEERING C35

Hybrid Built-in Self-Test

Methods and Tools for Analysis and Optimization of BIST

ELMET ORASSON

TUT
Press

TALLINN TECHNICAL UNIVERSITY
Faculty of Information Technology
Department of Computer Engineering
Chair of Computer Engineering and Diagnostics

Dissertation was accepted for the defense of the degree of Doctor of Philosophy in Computer and Systems Engineering on 5. October, 2007

Supervisor:	Prof. Dr. Raimund Ubar
Opponents:	Prof. Einar Aas, Norwegian University of Science and Technology (NTNU) in Trondheim, NORWAY
	Prof. Bernd Straube, Fraunhofer Institute for Integrated Circuits, Dresden, GERMANY
Commencement:	October 24, 2007

Thesis is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Engineering at Tallinn Technical University. Hereby I declare that this doctoral thesis, my own original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for any degree or examination.

/Elmet Orasson/

ISSN 1406-4731
ISBN 978-9985-59-730-9

INFORMAATIKA JA SÜSTEEMITEHNIKA C35

Sisseehitatud Hübriidne Isetestimine

Meetodid ja vahendid analüüsiks ning optimeerimiseks

ELMET ORASSON

TTÜ
Kirjastus

To my mother,

To my colleagues and friends

Abstract

Rapid advances in deep submicron and nanotechnologies, as well as in design automation are enabling engineers to design more complex integrated circuits (IC) and driving them toward new design paradigms like System-on-Chip (SoC) and Network-on-Chip (NoC). Such a design style allows to reuse previous designs and lead to shorter time to market and reduced cost. On the other hand, SoC design philosophy makes external test increasingly difficult. Internal speed of SoC is constantly increasing and the technology used in external testers is always one step behind. Therefore Built-In Self-Test (BIST) has emerged as a promising solution to the VLSI and SoC testing problem.

This thesis is dedicated to investigations and developments to improve the efficiency and quality of BIST architectures. First, hybrid BIST architectures, where pseudorandom test patterns are combined with deterministic test patterns, were researched. Then, a new functional hybrid BIST approach was developed where instead of pseudorandom test patterns the normal functional routines carried out in digital systems are combined with deterministic test patterns. And finally, a Design-for-Testability (DfT) technique was combined with BIST for sequential circuits to achieve the needed test quality.

A method and algorithms were developed for fast calculation of the cost of hybrid BIST. The cost model was used to find an optimal balance between pseudorandom and deterministic test sets, and to perform the hybrid self-test with minimum cost of both, time and memory, and without losing in test quality. The functional hybrid BIST approach was investigated in the case of testing microprogrammed data-paths in digital systems. A method was proposed to find the trade-off between the functional test-routines and deterministic test parts.

A new generic functional fault model was proposed to increase the accuracy of evaluating the fault coverage of given test sets, and as a consequence to improve the quality of testing including BIST. The fault model allows to map the physical transistor-level defects on the higher macro logic level which reduces the complexity of fault simulation algorithms and helps to increase the speed of test quality analysis.

To prove the efficiency of the developed new methods, a lot of software tools and scripts were developed and integrated into a joint R&D and teaching environment with the purpose to improve the teaching quality. The environment is targeted to e-learning the topics like fault simulation, test generation, BIST, and fault diagnosis in digital circuits. Experimental part of the work demonstrated the feasibility of the developed theoretical ideas, and the advantages compared to known approaches.

Keywords: Physical Defects, Fault models, Fault Simulation, Test Generation, Pseudorandom Test, Deterministic Test, Built-In Self-Test, Fault Diagnosis.

Resümee

Kiire areng submikron- ja nanotehnoloogias ning elektroonikadisaini automatiseerimine võimaldavad inseneridel projekteerida üha keerukamaid integraalskeeme, mis on kokkuvõttes viinud uutele disainiparadigmadele nagu "süsteem-kiibil" ja "võrk-kiibil". Kiipsüsteemide kontseptuaalne sisu tähendab eelnevalt projekteeritud moodulite korduvkasutamist, mis võimaldab disainiprotsesside kiirendamist, kiiremat sisenemist turule ja väiksemaid disainikulusid. Samas aga on süsteemide keerukuse kasv teinud traditsioonilise välise testimise erakordselt raskeks. Kuna kiipsüsteemides töösagedus kiiresti kasvab, siis on välised testrid testitavatest objektidest kiiruse mõttes alati sammu võrra taga. Nendel põhjustel on ülisuurte integraalskeemide testimisprobleemi lahendusena järjest enam hakanud levima sisseehitatud isetestimise kontseptsioon.

Väitekiri on pühendatud istestivate arhitektuuride efektiivsuse ja kvaliteedi uuringutele ja väljatöötlustele parandamisele. Esiteks uuriti ja töötati välja hübriidseid isetestimislahendusi, kus pseudojuhuteste kombineeritakse deterministlike testsignaalidega. Teiseks töötati välja uus funktsionaalne hübriid-isetestimise kontseptsioon, kus pseudojuhutesti asemel kasutatakse süsteemi enda funktsionaalseid signaale ning kombineeritakse neid deterministlike testidega. Kolmandaks kasutati isetestimise kvaliteedi tõstmiseks ka testitavuse parandamise meetodeid.

Töötati välja meetod ja algoritmid hübriid-isetestimise maksumuse arvutamiseks. Uut hinnamudelit kasutati töös optimaalse vahekorra leidmiseks pseudojuhutesti ja deterministliku testi vahel, minimeerides nii testimise aega kui ka mälutarvet ette antud kitsendustel. Töötati välja optimeerimismeetodid ka funktsionaalse hübriidtesti arhitektuuride jaoks.

Rikete simuleerimise täpsuse tõstmiseks ja testimise, s.h. ka isetestimislahenduste kvaliteedi parandamiseks töötati välja uut tüüpi funktsionaalne rikete mudel. Uus mudel võimaldab teisendada suvalisi loogikaliselt kirjeldatavaid füüsikalisi defekte loogikatasandile ja seeläbi tõsta rikete analüüsi produktiivsust digitaalskeemides.

Uute meetodite efektiivsuse tõestamiseks töötati välja terve hulk tarkvaratööriistu, mis integreeriti ühtsesse keskkonda nii uurimistöö kui ka õppetöö efektiivsuse suurendamiseks. Keskkonda on võimalik kasutada e-õppe eesmärgil niisuguste disaini ja testi probleemide uurimisel ja omandamisel nagu rikete simuleerimine, testide genereerimine, isetestimine ja rikete diagnoos digitaalskeemides. Töö eksperimentaalosas, mille aluseks oli nimetatud keskkond, õnnestus demonstreerida uute meetodite teostatavust ja suuremat efektiivsust võrreldes seniste meetoditega.

Võtmesõnad: füüsikalised defektid, rikete mudelid, rikete simuleerimine, testide genereerimine, pseudojuhutest, deterministlik test, isetestimine, rikete diagnoos.

Acknowledgements

When I started my studies at Tallinn University of Technology (TUT) I did not even imagine I would finish these with Ph.D degree. I hoped to become a diploma computer engineer instead. What really happened is a completely different story – Raimund-Johannes Ubar was giving lectures covering digital test issues and in one course I showed him a little program I had developed and used to generate test sequences. For me it was just another tool to get job done but he saw more potential behind it. This tool (called 'bist') is now part of Turbo Tester package and is one key component in most of the experiments discussed in this thesis. He and Margus Kruus were also the main characters behind the fact I continued my studies.

Very special thanks to Raimund-Johannes Ubar. First, for leading me to this interesting research area more than 10 years ago, and secondly, for being a major motivator behind the completion of this thesis. Without his support it would definitely have taken much longer.

Most of this work presented here is part of group efforts on different topics. Therefore I would like thank:

Heinz-Dietrich Wuttke and his colleagues from TU Ilmenau who supported the work on Java based Applets for Teaching Digital Test and on a special web-site for hosting theoretical material, laboratory exercises and software for teaching digital test courses.

Witold Pleskacz and his colleagues from Warzaw University of Technology who provided the DefSim chip along with the accompanying measurement hardware and supported the development of DefSim educational software.

Many thanks to Maili Markvardt for her suggestions, they helped much when catching typos and improving overall readability. She was motivating me to finish this thesis, also.

My mother who supported me in any way imaginable through all these years I have spent at university.

I would also thank Tarmo Robal, Helena Kruus and all my friends and colleagues not mentioned above

List of publications

1. R. Ubar, E. Orasson, H.-D. Wuttke. Interactive Teaching Software "Introduction To Digital Test". 45th International Conference, Ilmenau (Germany), October 4-6, 2000, pp. 949-954.

2. R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke. Combining Learning, Training and Research in Laboratory Course for Design and Test. 7th Baltic Electronics Conference, Tallinn, October 8-11, 2000, 221-224.

3. R. Ubar, G. Jervan, Z. Peng, E. Orasson, R. Raidma. Fast Test Cost Calculation for Hybrid BIST in Digital Systems. Proc. of EUROMICRO Symposium on Digital Systems Design, Warsaw, September 4-6, 2001, pp. 318-325.

4. R. Ubar, E. Orasson, T. Evartson. Java Applet for Self-Learning of Digital Test Issues. 13th EAEEIE Conference, York, Great Britannia, April 8-10, 2002

5. R. Ubar. E. Orasson, H.-D. Wuttke. Internet-Based Software for Teaching Test of Digital Circuits. 23rd Int. Conf. on Microelectronics. Nis, Yugoslavia, May 12-15 2002, Vol.2, pp. 659-662.

6. 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.

7. R. Ubar, E. Orasson, T. Evartson. Self-learning tool for digital test. Proceedings of 2nd Int. Conf. "Distance learning – educational sphere of the XXI century", Minsk, Belarus, Nov. 26-28, 2002, pp. 36-38.

8. 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.

9. 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. Summaries. Sousse, Tunisia, March 26-28, 2003, CIT-6, pp. 134.

10. 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.

11. S. Devadze, R. Gorjachev, A. Jutman, E. Orasson, V. Rosin, R. Ubar. E-Learning Tools for Digital Test. In "Distance Learning – Educational Environment of the XXI Century", Minsk, 2003, pp. 336-342.

12. R. Ubar, N. Mazurova, J. Smahtina, E. Orasson, J. Raik. HyFBIST: Hybrid Functional Built-In Self-Test in Microprogrammed Data-Paths of Digital Systems. Int. Conference MIXDES, Szczecin, June 24-26, 2004, pp. 497-502.

13. J. Raik, E. Orasson, R. Ubar. Sequential Circuits BIST with Status BIT Control. Int. Conference MIXDES, Szczecin, June 24-26, 2004, pp. 507-510.

14. E. Ivask, A. Jutman, E. Orasson, J. Raik, R. Ubar, H.-D. Wuttke. Research Environment for Teaching Digital Test. 49. Int. Conf. IWK, Ilmenau, Germany, September 27-30, 2004, pp. 468-473.

15. H. Kruus, E. Orasson, T. Robal, R. Ubar. Investigating Defects in Digital Circuits by Boolean Differential Equations. The 4th International Conference "Distance Learning – Educational Sphere of XXI Century" (DLESC'04), Minsk, November 10-13, 2004, pp. 432-435.

16. V. Vislogubov, A. Jutman, H. Kruus, E. Orasson, J. Raik, R. Ubar. Diagnostic Software with WEB Interface for Teaching Purposes. Proc. of the 9th Biennial Baltic Electronics Conference, Oct. 3-6, 2004, Tallinn, pp. 255-258

17. A. Jutman, J. Raik, E. Orasson, R. Ubar. Overview of the Educational Tools developed in REASON. Workshop on Research and Training Action for System on Chip DesigN – REASON, Tallinn, May 21, 2005, 7 p.

18. R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke. Teaching Advanced Test Issues in Digital Electronics. 6th IEEE International Conference on Information Technology Based Higher Education and Training. July 7-9, 2005, Santo Domingo, pp. S2B-5 – S2B-10.

19. R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke. Teaching Advanced Test Issues in Digital Electronics. Summary. 6th IEEE International Conference on Information Technology Based Higher Education and Training. July 7-9, 2005, Santo Domingo, pp. 46-47.

20. R. Ubar, G. Jervan, H. Kruus, E. Orasson, I. Aleksejev. Optimization of the Store-and-Generate Based Built-in Self-Test. Info- ja kommunikatsioonitehnoloogia doktorikooli IKTDK aastakonverentsi kogumik. ISBN 9985-59-624-2. Jäneda, 12.-13. mai 2006, pp. 93-96.

21. R. Ubar, G. Jervan, H. Kruus, E. Orasson, I. Aleksejev. Optimization of the Store-and-Generate Based Built-in Self-Test. Baltic Electronics Conference. Laulasmaa, Oct. 2006, pp.199-202.

22. R. Ubar, A. Jutman, M. Kruus, E. Orasson, S. Devadze, H.-D. Wuttke. Learning Digital Test and Diagnostics via Internet. International Journal of Emerging Technologies in Learning. International Journal of Online Engineering, Vol.3, No.1, pp. 1-9, 2007.

23. H. Kruus, G. Jervan, E. Orasson, R. Ubar. Optimization of Memory-Constrained Hybrid BIST for Testing Core-Based Systems. IKTDK 2007 aastakonverents. Viinistu, Mai 11-12, 2007, pp.133-136.

24. G. Jervan, H. Kruus, E. Orasson, R. Ubar. Optimization of Memory-Constrained Hybrid BIST for Testing Core-Based Systems. IEEE 2nd International Symposium on Industrial Embedded Systems - SIES'2007. Lisbon, Portugal, 4-6 July 2007.

25. G. Jervan, H. Kruus, E. Orasson, R. Ubar. Hybrid BIST Optimization Using Reseeding and Test Set Compaction. Proc. of 10th EUROMICRO Conference on Digital System Design - DSD 2007, Lübeck, Germany, August 27 - 31, 2007.

Abbreviations

ATE	Automated Test Equipment
ATPG	Automated Test Pattern Generation/Generator
BCU	BIST Control Unit
BILBO	Built-in Logic Block Observer
BIST	Built-In Self Test
CA	Cellular Automata
CAD	Computer Aided Design
CBET	Combination of BIST and External Test
CLI	Command Line Interface
CPU	Central Processing Unit
CUT	Circuit Under Test
DfT	Design for Testability
HTTF	Hard-To-Test-Faults
HyFBIST	Hybrid Functional BIST
IC	Integrated Circuit
IP	Intellectual Property
ISCAS	International Symposium on Circuits and Systems
ITRS	International Technology Roadmap for Semiconductors
LBIST	Logic BIST
LFSR	Linear Feedback Shift Register
NoC	Network-on-Chip
ORA	Output Response Analyzer
PCB	Printed Circuit Board
PPB	Pseudorandom Pattern Block
PPB	Pseudorandom Pattern Blocks
PRG	Pseudorandom Generator
ROM	Read-Only Memory
RPRF	Random Pattern Resistant Faults
SAF	Stuck-At Fault
SoC	System-on-Chip
SWR-BIST	Segment Weighted Random BIST
TAM	Test Access Mechanism
TPG	Test Pattern Generator
TRA	Test Response Analyzer
VLSI	Very Large Scale IC
WPS	Weighted Pseudorandom Sequences

Table of Contents

1 Introduction

According to the Moore's law [26], the scale of integrated circuits (IC) has doubled every 18 months. Today, very large scale ICs (VLSI) with many millions of transistors are commonly used in computers and electronic applications. This is a result of continuously decreasing dimensions (feature size) of the transistors and interconnecting wires from tens of microns to tens of nanometers. The reduction in feature size increases the probability that a manufacturing defect in the IC will result in a faulty chip. Therefore, testing is required to guarantee fault-free products.

Electronic testing includes IC testing, printed circuit board (PCB) testing, and system testing at the various manufacturing stages, and also during system operation in the field. Testing is used not only to find out faulty chips, boards or systems, but also to improve the production yield by analyzing the causes of faults. Periodic testing in the field is performed to ensure fault-free system operation. Hence, VLSI testing is important to designers, product engineers, test engineers, manufacturers, and end-users.

Because of the diversity of VLSI defects, and high complexity of VLSI devices and systems it is difficult to generate tests for real transistor level physical defects. Higher level and more uniform fault models are needed for generating and evaluating a set of test patterns. Generally, a good fault model should satisfy two criteria: it should accurately reflect the behavior of defects, and it should be computationally efficient and simple. The stuck-at fault model has been used already many decades as the basis for test generation and fault analysis. However, there are many other defect types like transistor faults, also referred as stuck-opens or stuck-shorts, open and short faults, different types of bridging faults between wires, delay faults, crosstalk faults, pattern sensitivity and coupling faults, analog faults etc. [50]. These faults are not well covered by the commonly accepted stuck-at fault model. Therefore research is going on to find better ways for modelling as well as describing existing physical defects.

The rapid advances in the areas of deep submicron and nanotechnologies, as well as in design automation are enabling engineers to design more and more complex integrated circuits. These developments are driving engineers toward new design paradigms like System-on-Chip (SoC) and Network-on-Chip (NoC). SoC is usually designed by embedding pre-designed complex functional blocks,

referred as cores, into one single die. Such a design style allows designers to reuse previous designs and will lead to a shorter time to market and a reduced cost. Testing of SoC, on the other hand, shares all the problems related to testing modern deep submicron chips, and introduces also some additional challenges due to the protection of intellectual property as well as the increased complexity and higher density [17].

Test engineers usually have to develop test sets after the design is completed. This requires a lot of time and effort that could be avoided if testing was considered already during the early design phase to make the design better testable.

As a result, a new research field was established called design for testability (DfT). DfT techniques generally fall into the following three categories: *ad hoc* DfT techniques, scan design techniques, and built-in self-test (BIST) techniques.

SoC design philosophy makes external testing increasingly difficult. Also, the internal speed of SoC is constantly increasing and the technology used in external automated test equipments (ATE) is always lagging one step behind. Therefore, the built-in self-test has emerged as a promising solution to the VLSI and SoC testing problem.

The requirement to integrate an increasing number of functions on a single chip, giving rise to system-on-chip (SoC) designs, has made the problem of testing them more difficult. Not only the fault models for various logic and memory functions are diverse, there is also restricted observability and controllability for individual embedded modules or cores in a SoC. This significantly impacts the test cost and test quality of such SoCs. The cost of test is affected by many factors like - test application time, independence on external tester infrastructure, dedicated access to embedded IP, stress mode and normal mode test quality, field test, etc., commonly addressed by BIST techniques [102].

BIST is a design for testability methodology aimed at detecting faulty components in a system by incorporating test logic on-chip [50]. In BIST, a portion of a circuit on a chip, board or system is used to test the digital circuit itself. Depending on the tested circuit type Logic BIST and Memory BIST techniques are used [50]. In this work, only Logic BIST is considered. In the following text we skip the attribute “Logic” and use only the term “BIST”, however, still meaning “Logic BIST”.

BIST was introduced already two decades ago [14, 73] to integrate a test pattern generator (TPG) and an output response analyzer (ORA) in the VLSI device to perform internal IC testing as illustrated in Fig 1-1.

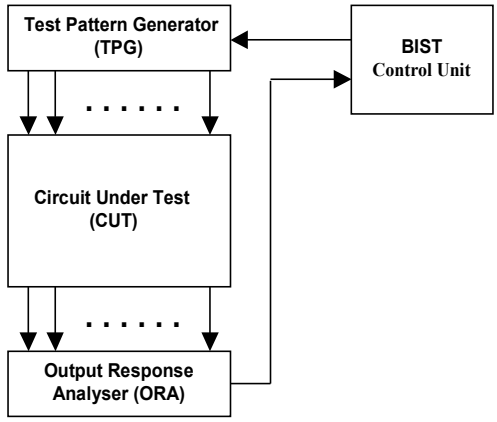


Fig 1-1. Classical built-in self-test architecture

The TPG automatically generates test patterns for application to the inputs of the CUT. The ORA automatically compacts the output responses of the CUT into a signature. Specific BIST timing control signals are generated by the BIST control unit for coordinating the BIST operation among the TPG, CUT and ORA. The BIST Control Unit also provides pass/fail indication once the BIST operation is complete. It includes comparison logic to compare the final signature with an embedded expected (i.e. known "good") signature.

In traditional BIST architectures, test pattern generation is mostly performed by *ad hoc* circuitry, typically Linear Feedback Shift Registers (LFSR) [9], cellular automata [71] or multifunctional registers like BILBO (Built-in Logic Block Observer) [59]. These circuitries are used to generate test sequences for exhaustive, pseudo-random, and pseudo-exhaustive testing. Exhaustive testing always guarantees 100% single-stuck and multiple-stuck fault coverage efficiency. On the other hand, exhaustive test can take too much time for combinational circuits with a lot of inputs. Therefore, pseudo-random testing [72] is often used for generating a subset of the exhaustive test. In this case, fault simulation is needed to calculate the exact fault coverage.

There are a number of advantages to use BIST techniques like:

- no ATE is needed
- allows at-speed testing
- the cost is reduced due to the reduced test time, and tester memory requirements.

However, there are also disadvantages associated with this approach, and related either to long test or to the insufficient fault coverage. Many circuits contain random-pattern-resistant faults which limit the fault coverage achieved by pseudorandom test patterns alone. To overcome the mentioned disadvantages, different modifications of BIST like hybrid BIST or improvements of BIST by DfT techniques have been introduced.

In hybrid BIST, also called as mixed-mode BIST [50], the pseudorandom test is combined with deterministic test data in different ways by using methods like ROM compression [107], LFSR reseeding [8], embedding deterministic patterns [67] and other methods discussed below in the next chapter. The task of combining pseudorandom and deterministic test data leads to a difficult combinatiorial problem of finding an optimal trade-off between pseudorandom and deterministic test data or minimal test cost under given constraints.

These optimization tasks are also very closely related to another NP-complete task of generating optimized deterministic test sets. The simplest approach to hybrid BIST is to perform top-up automated test pattern generation (ATPG) for the faults not detected by pseudorandom BIST. It helps to obtain a supplementary subset of deterministic test patterns that "top-up" the fault coverage to the desired level and then store this part of the test set directly on the tester. This desired level of fault coverage should be reached at minimal test cost.

The work under this thesis was devoted to development of methods and algorithms of finding optimal trade-offs between pseudorandom and deterministic test parts in the hybrid BIST under different constraints. Investigation of the problem how to use DfT to improve the quality of BIST solutions was also one of the research topics of this work. Both of these research topics were closely related to the tasks of deterministic test generation and fault simulation. In such a way a very broad scope of test problems had to be covered. Consequently, to carry out all the investigations in this thesis, a lot of tools had to be developed or accommodated and integrated into different workflows for performing experimental proof of feasibility or efficiency of the proposed algorithms and methodological solutions.

This research work also involved students were in form of tasks for course or diploma works. The experience of working with students gave an idea of setting up an e-learning environment presenting tools for modelling the processes of test generation, fault simulation, pseudorandom test pattern generation, fault diagnosis, BIST cost estimation, BIST quality evaluation and hybrid BIST optimization.

Our society is becoming increasingly dependent on computing systems. This dependency is especially felt upon the occurrence of failures in systems. That is the reason why design for dependability and test are becoming more and more important in all the life periods of electronic systems, and therefore, these issues should be considered also in teaching tomorrows system engineers. Students and young engineers should be trained on integrating design and test solutions. Teaching in this domain should be facilitated by using integrated CAD tools that support design verification, testability analysis, design for testability, test generation, fault simulation, built-in self-test, fault diagnosis and fault tolerance.

Tools developed for the research were also integrated into an e-Learning and research environment as means for improving digital test teaching at technical universities in the field of hybrid BIST. This can be regarded as a side-effect of the experimental research and should help to prove the feasibility and correctness of new ideas presented here.

2 State-of-the-art

2.1 Fault models

Testing and diagnosis of digital electronics systems have faced a lot of problems produced mainly by the complexity of systems. The efficiency of test generation (quality of tests, speed of test generation) is highly dependent on the system description and chosen fault models.

Many fault models have been proposed, like:

- transistor stuck-opens and stuck-shorts
- gate open and short faults
- different types of bridging faults between wires
- delay faults
- crosstalk faults
- pattern sensitivity and coupling faults
- analog faults, etc [50, 59]

Unfortunately, no single fault model accurately reflects the behavior of all possible defects. Often, a combination of different fault models is used, making test generation, fault analysis and test quality evaluation difficult. As a compromise, simple gate-level stuck-at fault model has been chosen as a representative fault model for all possible physical defects, and it has been used for decades in the industry as the target for test generation and as the unit of measure in test quality evaluation.

Since traditional low-level test generation methods and tools for complex VLSI systems have lost their importance, other approaches based mainly on functional, behavioral, or hierarchical methods are gaining more and more popularity [15, 98]. The advantage of hierarchical test generation approaches compared to the functional ones lies in the possibility of constructing test plans at higher functional levels and modeling faults at lower levels.

Conventional low-level fault modeling methods use logic gate level representations of digital circuits and stuck-at-0/stuck-at-1 fault (SAF) model. The rather popular SAF model in test quality estimation has not withstood the test of time. It has been shown that high SAF coverage cannot guarantee high quality of testing, for example, for CMOS integrated circuits [43, 44, 109]. The

reason is that the SAF model ignores the actual behavior of digital circuits implemented as CMOS integrated circuits, and does not adequately represent the majority of real IC defects and failure mechanisms.

Physical defects occurring in real-life circuits often do not manifest themselves as stuck-at faults. Moreover, the types of faults that can be observed in a real gate depend not only on the logic function of the gate, but also on its physical design. Although these facts are well known, in engineering practice they are usually ignored. A layout-based test generation technique proposed in [74] is limited to current testing methodology. Another work [62] treats the probabilities of defects in an oversimplified way.

To improve the test quality, there is a need to replace abstract fault models like SAF with more realistic defect models.

In this work, a uniform fault model for representing physical defects in components of digital circuits is introduced. Physical defects are modeled as parameters in generic Boolean differential equations. Solutions of the equations give the conditions at which defects are locally activated. The defect activation conditions are used as functional fault models on the logic level for fault simulation purposes.

Using the proposed new functional fault model improves the exactness of test quality evaluation in the BIST design methods developed in this thesis.

2.2 Hybrid BIST methods

Rapid advances of the microelectronics technology in recent years have brought new possibilities to the design and manufacturing of integrated circuits (ICs) [103]. Nowadays many systems are designed by embedding pre-designed and pre-verified complex functional blocks, usually referred as cores, into one single die. Such core-based design technique has led to increased design productivity, but at the same time it has introduced additional test-related problems. These additional testing problems, together with the test problems induced due to the complexity and heterogeneous nature of such systems-on-chip (SoC), pose great challenges to the SoC testing community [115].

Typically, a SoC consists of microprocessor cores, digital logic blocks, analogue devices, and memory structures [17]. These different types of components were traditionally tested as separate chips by dedicated automatic test equipment of different types. Now they must be tested all together as a single chip either by a

super tester, capable of handling different types of cores and is very expensive, or by multiple testers, which is very time-consuming due to the time of moving from one tester to another. Hence, philosophy of SoC design makes external testing increasingly difficult. Also, the internal speed of SoC is constantly increasing and the technology used in external automated test equipments (ATE) is always lagging one step behind.

For the reasons described above, the Built-In Self-Test (BIST) has emerged as a promising solution to the VLSI and SoC testing problem. BIST is a DfT methodology aimed at detecting faulty components in a system by incorporating test logic on-chip [115].

Another key issue to be addressed for SoC testing is the implementation of test access mechanisms on the chip. For traditional system-on-board design, direct test access to the peripheries of the basic components, in the form of separate chips, is usually available. For the corresponding cores, embedded deeply in a SoC, such access is impossible. Therefore, additional test access mechanisms must be included in a SoC to connect the core peripheries to the test sources and sinks, which are the SoC pins when testing by an external tester.

When using BIST the test access costs can be substantially reduced by putting test sources and sinks right next to the cores to be testes. This is one way of dealing with the discrepancy between rapidly increasing SoC and slow tester speeds. The testers have hard time to match typical SoC clock frequencies because of the simple fact - testers are built on previous technology level whereas SoCs represent the new level. The introduction of BIST mechanisms in a SoC will also improve the diagnostic ability and on-the-field/on-the-fly test capability, essential for applications where regular operation and maintenance testing is important [17].

One major concern with implementing the BIST is hardware overhead minimization, since test pattern generation and application is performed by additional on-chip harware. Unlike stored pattern BIST, which requires high hardware overhead due to required memory devices for storing precomputed test patterns, a pseudorandom BIST utilizes pseudorandom pattern generators such as linear feedback shift registers (LFSRs) [9], cellular automata (CA) [71] or multifunctional registers like BILBO (Built-in Logic Block Observer) [59] and, therefore, requires very little hardware overhead. However, LFSR or CA generated random patterns often require unacceptably long test sequences (and increasing test length means longer application time) for achieving high fault coverage for CUTs that contain many random pattern resistant faults (RPRFs). It has been shown that only a few RPRFs are often determining the random pattern test length [72], required for achieving high fault coverage. Usually, complex

circuits contain many RPRFs [16] which in turn limits the fault coverage that can be achieved with pseudorandom BIST approach.

One method for improving the fault coverage for a test-per-scan BIST is to modify the CUT by either inserting test points [68] or by redesigning it to improve the fault coverage [114]. The drawback of these techniques is that they generally add additional logic to the circuitry, degrading system performance.

Some other approaches are using weighted pseudorandom sequences (WPS) for BIST fault coverage improvement. In these approaches additional weight control logic is needed to change the probabilities of each bit in the test sequence. In weighted random pattern testing [29, 42, 45, 72], the outputs of test pattern generator (TPG) are biased to generate test sequences that have non-uniform signal probabilities to increase detection probabilities of RPRFs that escape pseudorandom test sequences which have a uniform signal probability of 0.5. Random pattern generators proposed in [69, 108] use Markov sources to exploit spatial correlation between state inputs that are consecutively located in the scan chain.

The weight logic can be placed either at the input of the scan chain [61] or in the individual scan cells themselves [8]. The disadvantage of the WPS approach is that the weight sets have to be stored on chip and additional control logic is required to switch between different weights. Therefore, the hardware overhead may become large.

A third method for BIST fault coverage improvement is to use dual stage test application. In the first test phase LFSR generated pseudorandom patterns, detecting the random pattern testable faults. Thereafter some additional deterministic test patterns are applied for detecting remaining random pattern resistant faults. This approach is called also as “mixed mode” BIST [50]. This alternative might be as effective as WPS in terms of test length but requires substantially less additional hardware.

In case of many stored deterministic patterns this may cause large hardware overhead. Overcoming this drawback of hybrid BIST requires additional, different methods for compressing deterministic test data like ROM compression, LFSR reseeding, embedding deterministic patterns, etc.

The simpliest way for generating deterministic test patterns on-chip is to store them in a Read-Only Memory (ROM). The disadvantage of this approach is related to the size of required ROM which can be unacceptably high in case of very complex CUTs. Several ROM compression techniques have been proposed for reducing storage size [25, 60, 90, 107].

The idea of LFSR reseeding is to compute a set of seeds, that when expanded by the LFSR, will produce the pre-generated deterministic test cubes. So, instead of storing complete test sequence, a much smaller amount of LFSR seeds is needed. These seeds can be computed by linear algebra as described in [8], for example. Because the seeds are much smaller than the test patterns, they also require less ROM storage.

One problem is that for an LFSR with a fixed characteristic (feedback) polynomial, it may not always be possible to find the proper seeds that can lead to the needed fault coverage. This problem was solved in [96, 97] by introducing a multi-polynomial LFSR where a possibility for choosing between different feedback polynomials was introduced when encoding a test cube.

Techniques for further reductions in storage of deterministic test patterns can be achieved by using variable-length seeds [41, 65], a special ATPG algorithm using concatenation of test cubes [94], folding counters [30], and seed encoding [1].

Another approach for mixed-mode BIST is to embed the deterministic test patterns in the pseudo-random sequence. One possible way to do this is to transform the "useless" pseudorandom test patterns which detect no new faults into the needed deterministic patterns. This idea has been implemented by adding mapping logic between the scan chains and the CUT [67], and by adding the mapping logic at the inputs to the scan chains to either perform bit-fixing [66] or bit-flipping [24, 33] techniques. The drawback of the described approaches is that the BIST architecture is extremely tailored to the specific circuit, and any change in the test set requires re-synthesis of the complete BIST hardware.

For manufacturing fault coverage enhancement where an external tester is present, deterministic data from the tester can be used to improve the fault coverage. The deterministic patterns stored in the external tester are coded in a compressed form and the BIST hardware inside the chip is then used to decompress it. Such techniques are described in [2, 12, 48, 76].

In a SoC, test scheduling can be done to overlap the BIST run time with the transfer time for loading the deterministic test patterns from the tester [22, 57].

Although the attainment of high fault coverage with practical lengths of test sequences is still one major concern of BIST techniques, reducing switching activity has become another important objective. It has been observed that switching activity during test application is often significantly higher than that

during the normal operation [111], therefore special caution must be exercised not to overload the CUT. In [100] a low hardware overhead TPG is presented for scan based BIST that can reduce switching activity in CUTs during BIST and also achieve very high fault coverage with a reasonable length of test sequence. In [11] a segment weighted random BIST (SWR-BIST) technique for low power testing is presented. The technique divides the scan chain into segments of different weights whereas heavily weighted segments have more biased probability than lightly weighted segments, and heavily weighted segments are placed closer to the end of scan chain than the lightly weighted segments, so the scan-in transitions are minimized.

In [22, 88], a trade-off problem between the cost of memory or power consumption and the test length in hybrid BIST, which consists of two parts pseudorandom test and deterministic test, was investigated.

In [101] the problems of optimizing a combined BIST and Automated Test Equipment (ATE) process are considered to meet the high fault coverage while preserving acceptable costs. For digital systems, the costs associated with a combined BIST/ATE testing process mainly consist of the following components:

- the cost due to the BIST area overhead and
- the cost due to the overall test time.

In general, BIST is faster than ATE, but it can provide only a limited fault coverage. For attaining a higher fault coverage from BIST, additional area (at a corresponding higher cost) is required. However, a higher fault coverage can usually be achieved from ATE, but excessive use of ATE results in additional test time (as an increased cost). The fault coverage of BIST and ATE plays a significant role, because it can affect the area overhead in BIST and the test time in BIST/ATE. In [101] a novel numerical method is proposed to find the optimized fault coverage by BIST and ATE so that minimal cost can be achieved.

A lot of papers have been devoted to using BIST for diagnosis purposes [5, 10, 27, 32, 38, 53, 52, 87]. Some papers are concentrating on BIST targeting specific fault models like [113] for delay faults and [47] for interconnection faults testing. In [37, 99, 110] BIST dependability issues are researched. Hierarchical BIST methods are considered in [6, 34]. In [6] a complete hierarchical framework is presented for BIST scheduling, data delivering, and diagnosis of a complex system including embedded cores with different test requirements. In [34] a hierarchical BIST method is proposed for testing a SoC with a global BIST controller, multiple local BIST circuits for each macro, and data/control paths to perform the SoC test operations.

The task of optimizing hybrid BIST leads to a difficult combinatorial problem of finding optimal trade-offs between pseudorandom and deterministic test data, or minimizing the test cost under given constraints. As the complexity of design and fabrication highly integrated systems on chip rapidly increases, the optimization of hybrid BIST remains a major concern of BIST techniques.

The main research objective in this thesis is to investigate and develop alternative methods for optimizing hybrid BIST by finding optimal balance between pseudorandom and deterministic test pattern sets, whereas the deterministic patterns may be used as seeds for LFSR. The optimization methods and algorithms are based on the store-and-generate technique as the general scheme with special cases and the optimization criteria is to minimize the test cost at given constraints on the fault coverage and the cost of the memory that is needed for storing deterministic test data.

2.3 E-Learning in Digital Test

The increasing complexity of digital systems accompanied by entering the era of Systems-in-Chips (SoC) and Networks-on-Chips (NoC) has made testing and fault diagnosis in electronic systems one of the most complicated and time-consuming problems in electronics design and manufacturing. Because of the very high cost of testing electronic products and the increasing complexity of the electronics systems, the importance of DfT is growing steadily.

Recent reviews have discovered that most VLSI and system designers know little about testing and design for testability (DfT) of today's digital systems because of the gap in education. In the today's university curricula test issues are usually neglected: students learn how to design electronic systems but not how to test them. The importance of test and fault diagnosis as a teaching objective is underestimated in traditional engineering education [106].

In a design course test is usually taught as a subtopic of minor importance an is generally taught as an independent discipline only when it is a "hobby horse" of a professor. There are two reasons for that. The first one is because the test is interpreted as a nonproductive (read: not important) issue compared to design. The second one can be explained by so called Tenhunen's Law which claims that the number of courses that should be taught at universities doubles in a decade [28]. To select courses for curricula is a difficult issue. And often a test course as a component of engineering education is left outside the curricula because of tough competition between courses.

On the other hand, the topics like Testing and Fault Diagnosis are not only issues related to Electronics Systems, they have an important didactive role for the engineering education in general [70]. First, testing is a method to learn how to ask right questions, second, it develops the ability to analyze cause-effect relationships, and third, diagnosis is looking for answers to the questions like "what is the reason of that what happened?"

Logic world (computers, digital circuits and systems, SoC) because of its inherent logical complexity could be the best objective for learning the concepts of testing and diagnostic analysis for any type of system in general. The real targets of education are: creativity, critical thinking, problem solving skills. Therefore, learning testing at a university should be research-oriented.

Moving towards multi-million gate System-on-Chips (SoC) makes embedded test 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 will get hands-on experience in solving test problems in digital systems like SoC.

In this thesis a teaching/learning environment is presented with the purpose to increase the teaching quality in the field of electronics DfT by hands-on training exercises. The problems of test generation, fault simulation, fault diagnosis and optimized BIST design are covered. The environment supports basic learning and advanced research oriented laboratory training. The interactive modules are focused on easy action and reaction, learning by doing, a game-like use, and encouraging students critical thinking, problem solving, and creativity.

3 Defect Modelling in BIST

As the complexity of digital systems continues to increase, the traditional low level methods for test generation and fault simulation have become obsolete. Other approaches based mainly on higher level functional and behavioral methods are gaining more popularity.

However, the trend towards higher level modeling 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. On the other hand, the defect-orientation is increasing even more the complexity.
To get out from the deadlock, the two opposite trends – high-level modeling and defect-orientation – should be combined into hierarchical approaches. The advantage of hierarchical approaches compared to high-level functional modeling lies in the possibility of constructing test plans on higher levels, and modeling faults on more detailed lower levels.

In this Chapter a new fault model is proposed to map the physical defects from the very low transistor level to gate or higher macro-gate level. The results of this chapter are published in [35, 82, 91].

3.1 Modelling Defects by Boolean Differential Equations

In the following, an approach is presented to model physical defects by generic Boolean differential equations with the goal to map them from physical level to logic level. A new physical defect oriented fault model is defined on that basis called functional fault model.

Consider a Boolean function $y = f(x_1, x_2, \ldots, x_n)$ implemented by an embedded component C in a circuit. Introduce a Boolean variable d for representing a given physical defect in the component, which may affect the value y by converting the Boolean function f into another function

$$y = f^d(x_1, x_2, \ldots, x_n)$$

where in fact, some of the arguments x_i can fall out, simplifying in that way the function because of the fault.

For the block C, let us introduce a generic parametric function

$$y^* = f^*(x_1, x_2, \ldots, x_n, d) = \overline{d}\, f \vee d\, f^d \tag{3-1}$$

as a function of a defect variable *d*, which describes the behavior of the component simultaneously for both possible fault-free and faulty cases. This could be written in a different manner:

$$y^* = \begin{cases} f, & if\ d = 0 \quad (faultfree) \\ f^d, & if\ d = 1 \quad (faulty) \end{cases} \tag{3-2}$$

The solutions of the Boolean differential equation

$$W^d = \frac{\partial y^*}{\partial d} = 1 \tag{3-3}$$

describe the conditions which activate the defect *d* on a line *y*. The parametric modeling of a given defect d by equations (3-1) and (3-3) allows us to use the constraints $W^d = 1$, either in defect-oriented fault simulation, for checking if the condition (3-3) is fulfilled, or in defect-oriented test generation, to solve the equation (3-3) when the defect d should be activated and tested.

To find W^d for a given defect *d* we have to create the corresponding logic expression for the faulty function f^d either by logical reasoning, by carrying out directly defect simulation, or by carrying out real experiments to learn the physical behavior of different defects.

Example 1.

Let us have a transistor circuit in Fig. 3-1 which implements the function $y = \overline{x_1 x_2 x_3 \vee x_4 x_5}$. A short defect as shown in Fig. 3-1 changes the function of the circuit as follows: $y^d = \overline{(x_1 \vee x_4) \wedge (x_2 x_3 \vee x_5)}$. Properties of Boolean differential allow removing extra inversion from the given formulas.

Thus, using the defect variable d for the short, we can create a generic differential equation for this defect and simplify the created expression as follows:

$$\frac{\partial y^*}{\partial d} = \frac{\partial\left[(x_1 x_2 x_3 \vee x_4 x_5)\overline{d} \vee (x_1 \vee x_4) \wedge (x_2 x_3 \vee x_5) d\right]}{\partial d} =$$

$$= x_1 \overline{x_2}\,\overline{x_4} x_5 \vee x_1 \overline{x_3}\,\overline{x_4} x_5 \vee \overline{x_1} x_2 x_3 x_4 \overline{x_5} = 1$$

From the equation three possible solutions follow: T = {10x01, 1x001, 01110}. Each of them can be used as a test pattern for the given short. On this contra-example, it is easy to show the inadequacy of the stuck-at fault (SAF) model for testing the transistor level faults. For example, the set of five test patterns 1110x, 0xx11, 01101, 10110, 11010 which test all the stuck-at faults in the circuit does not include any of the possible test solutions for detecting the short from the set T.

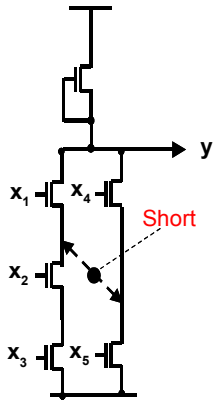


Fig. 3-1. Transistor circuit with a short

Note, that for the same purposes of finding the test for the defect d we could solve also directly the equation

$$f \oplus f^d = (x_1 x_2 x_3 \vee x_4 x_5) \oplus (x_1 \vee x_4)(x_2 x_3 \vee x_5) = 1 \qquad (3\text{-}4)$$

without introducing the defect variable d. However, solving the equation (3-3) will be much easier than (3-4) because of simplification possibilities resulting from specific properties of Boolean differentials [4].

3.2 Mapping Physical Transistor Defects to Logic Level

The described method represents a general approach to map an arbitrary physical defect onto a higher (in this case, logic) level. By the described approach an arbitrary physical defect in a component can be represented by a logical constraint $W^d = 1$ to be fulfilled for activating the defect (Fig. 3-2).

The event of erroneous value on the output y of a functional component can be described as $dy = 1$, where dy means Boolean differential. A functional fault representing a defect d can be described as a couple (dy, W^d). At the presence of a physical level defect d, we will have an higher level erroneous signal $dy = 1$ iff the condition $W^d = 1$ is fulfilled.

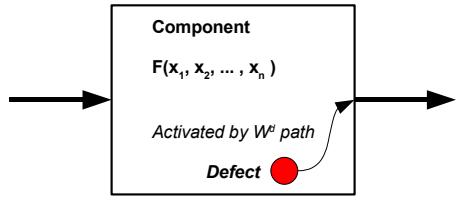


Fig. 3-2. Modelling a physical defect by a logic constraint

From another point of view, the equation (3-3) can be interpreted as a mapping of a physical defect d from the transitor level to the logic level as an erroneous change of a logic value $dy = 1$ by means of fulfilling the logic condition $W^d = 1$. The following examples will show the feasibility of using Boolean differential equations for mapping faults from physical transistor level to the logic level.

Example 2.

Transistor level stuck-on faults. The behavior of the transistor level NOR gate depicted in Fig. 3-3 cannot be described strictly logically.

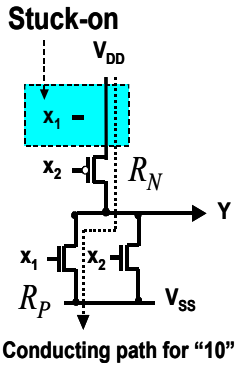


Fig. 3-3. Stuck-on fault in the transistor NOR gate

The input vector "10" produces a conducting path from V_{DD} to V_{SS}, and the corresponding voltage at the output node Y will not be equal to either V_{DD} or V_{SS} but will instead be a function of the voltage divider formed by the channel resistances of the conducting transistors:

$$V_Y = \frac{V_{DD} R_N}{(R_P + R_N)}$$

Depending on the ratio of these resistances along with the switching thresholds of the gates being driven by the output of the faulty gate y, the output voltage of the faulty gate may or may not be detected at a primary output. This ambiguous value on the gate output will be denoted by introducing the variable *Z*. Faulty function of the gate can then be represented as follows:

$$y^d = \overline{x_1 x_2} \vee x_1 \overline{x_2} Z$$

If $x_1 \overline{x_2} = 1$ then $y^d = Z$ Using now the expressions (3-1) and (3-3) we get:

$$y^* = \overline{d}\,(\overline{x_1 \vee x_2}) \vee d\,(\overline{x_1 x_2} \vee x_1 \overline{x_2} Z)$$

$$W^d = \frac{\partial y^*}{\partial d} = x_1 \overline{x_2} Z = 1$$

From that it follows that the condition to activate the defect is $x_1 = 1, x_2 = 0$

Example 3.

Transistor level stuck-open faults. For the transistor stuck-open fault of the NOR gate in Fig. 3-4, there will be no path from the output node to either V_{DD} or V_{SS} for some input patterns. As a result, the output node will retain its previous logic value. This creates a situation where a combinational logic gate behaves like a dynamic memory element.

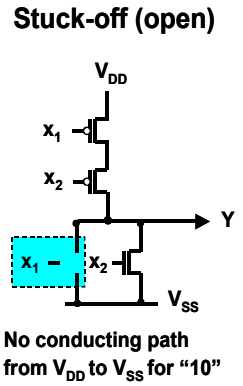


Fig. 3-4. Stuck-off (open) fault in the transistor NOR gate

The faulty function of the gate is: $y^d = \overline{x_1 x_2} \vee x_1 \overline{x_2}\, y'$ where y' corresponds to the output value stored at the output of the faulty gate. Using now the expressions (3-1) and (3-3) we get:

$$y^* = \overline{d}\,(\overline{x_1 \vee x_2}) \vee d(\overline{x_1 x_2} \vee x_1 \overline{x_2}\, y')$$
$$W^d = \frac{\partial y^*}{\partial d} = x_1 \overline{x_2}\, y' = 1$$

From that it follows that the condition to activate the defect is $x_1 = 1, x_2 = 0, y' = 1$. In other words, for testing the fault we need a test sequence of two patterns: "00" to get the value 1 on the output, and then "11".

Some examples of the conditions W^d for different type of gate-level defects (where stuck-at-fault (SAF) is a particular case) are given in Table 3-1 (here x_k is the observable variable, and x'_k is the variable at the previous time moment).

Table 3-1. Examples of representing typical gate-level faults by the condition W^d

#	Defect	Conditions W^d
1	SAF $x_k \equiv 0$	$x_k = 1$
2	SAF $x_k \equiv 1$	$x_k = 0$
3	Short between x_k and x_l	$x_k = 1, x_l = 0$
4	Exchange of lines x_k and x_l	$x_k = 1, x_l = 0$; $x_k = 0, x_l = 1$
5	Delay fault on the line x_k	$x_k = 1, x'_k = 0$; $x_k = 0, x'_k = 1$

3.3 Mapping Interconnection Defects to Logic Level

Consider now a component C representing a Boolean function $y = f(x_1, x_2, \ldots, x_n)$ embedded in an environment given by a subset of lines $E_c = x_{n+1}, \ldots, x_p$. Introduce the same Boolean variable d for representing physical defects in the subcircuit (C,E_c), given by the block C with its neighborhood E_c, which may affect on the value y. Let the defect d convert the Boolean function f into another function

$$y = f^d(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_p)$$

Introduce for modelling physical defects related to the subcircuit (C,E_c) a generic parametric function

$$y^* = f^*(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_p, d) = (\overline{d} \wedge f) \vee (d \wedge f^d)$$

as a function of a defect variable d, which describes the behavior of the subcircuit simultaneously for both, fault-free and faulty cases. For the faulty case the value of the defect variable d as a parameter is equal to 1, and for the fault-free case d = 0. In other words, $y^* = f^d$ if $d = 1$, and $y^* = f$ if $d = 0$. The solutions of the Boolean differential equation (3-3) describe the conditions which activate the defect d on a line y.

Example 4.

A short between two lines x_k and x_l in the circuit (Fig. 3-5). The faulty function of x_k in the case of the defect d in accordance to the wired-AND fault model can be represented as .

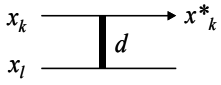


Fig. 3-5. A bridging fault

Introduce now a generic parametric function

$$x_k^* = f(x_k, x_l, d) = \overline{d}\, x_k \vee d\, x_k^d = \overline{d}\, x_k \vee d\,(x_k x_l)$$

as a function of a defect variable d, which describes the behavior of the interconnection network simultaneously for both, fault-free and faulty cases. The solution of the Boolean differential equation

$$W^d = \frac{\partial y^*}{\partial d} = x_k \overline{k_l}$$

describes the conditions (constraints) which activate the fault d on a line x_k (Fig. 3-5). The condition means that in order to detect the short between lines x_k and x_l we have to assign to x_k the value 1 and to x_l the value 0.

Example 5.

A short between two lines x_k and x_l in the circuit which creates a feedback loop. A circuit with such a loop and its equivalent faulty circuit corresponding to the wired-AND fault model is depicted in Fig. 3-6.

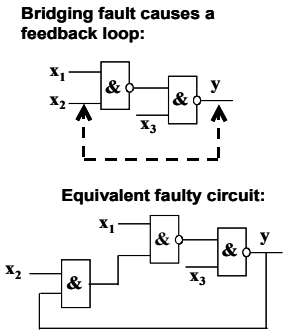


Fig. 3-6. Bridging fault with feedback loop

The generic parametric function for describing the behavior of the circuit simultaneously for both, fault-free and faulty cases has the following form:

$$y^* = \overline{d}(x_1 x_2 \vee \overline{x_3}) \vee d(x_1 x_2 y' \vee \overline{x_3}) = x_1 x_2 (\overline{d} \vee y') \overline{x_3}$$

The solution of the Boolean differential equation

$$W^d = \frac{\partial y^*}{\partial d} = x_1 x_2 x_3 \overline{y'} = 1$$

describes the conditions (constraints) which activate the fault d on a line y (Fig. 3-6). The apostrophe at y means that the value of y belongs to the previous time moment. The condition $W^d = x_1 x_2 x_3 \overline{y'} = 1$ means that a sequence of two patterns is needed for the testing of this short. First, we have to set $y = 0$, by assigning $x_3 = 0$ for example, then we have to apply the pattern $x_1 = 1, x_2 = 1, x_3 = 1$.

From this example we see that the constraints for activating a fault may be spread over different time moments, and represent a sequences of patterns. We see also that the method for describing faults by generic Boolean differential equations allows us directly to attack the problem of testing so called "sequential faults" which convert combinational circuits into sequential ones, or which increase the number of states in sequential circuits. Test generators which are able to work with such faults are missing.

The functional fault model described as a couple (dy, W^d) can be regarded first, as a method of mapping arbitrary physical defects onto the logic level, and second, as a universal method of fault modeling in hierarchical approaches to test generation and fault simulation.

The conditions W^d for activating defects d can be used as constraints at the higher (logical or register transfer) level either for fault simulation or for test pattern generation without paying attention to the physical reasons of defects.

3.4 Conclusions

1. An approach is presented to map physical defects in digital circuits and systems from transistor level to logic level for test generation and fault simulation purposes.

2. For modelling physical defects generic Boolean differential equations were introduced which allow to map the physical faults from lower physical level to higher logic level.

3. Different transistor level defects were analyzed to show that this way of mapping is general and feasible.

4. A new fault model was defined on that basis, called functional fault model.

5. The functional fault model can be regarded as a uniform interface for mapping faults from a given arbitrary level to the next higher level.

4 Hybrid Built-In Self-Test

In this chapter a hybrid BIST solution is developed for testing systems-on-chip (SoC), which combines pseudorandom test patterns with stored precomputed deterministic test patterns. A procedure is proposed for fast calculation of the cost of hybrid BIST at different lengths of pseudorandom test to find an optimal balance between test sets, and to perform core test with minimum cost of both, time and memory, and without losing in test quality. Compared to the known approaches, based on iterative use of deterministic ATPG for evaluating the cost of stored patterns, in this paper a new, extremely fast procedure is proposed, which calculates costs on a basis of fault table manipulations. Experiments on the ISCAS benchmark circuits show that the new procedure is about two orders of magnitude faster than the previous one.

The results of this chapter are published in [21, 20, 83, 84].

4.1 Principles of Hybrid BIST

To test the electronic system we need test pattern source and sink together with an appropriate test access mechanism (TAM) [112]. Such a test architecture can be implemented in several different ways. A widespread approach implements both source and sink off-chip and requires therefore the use of external Automatic Test Equipment (ATE). But rapid advances in recent years have enabled the integrated circuits (ICs) manufactures to move towards very deep submicron technologies and to integrate several complex functional blocks into one single chip. The internal speed of such a Systems-on-Chip (SoC) is constantly increasing but the technology used in ATE is always one step behind and therefore the ATE solution has already become unacceptably expensive and inaccurate [103]. Therefore, in order to apply at-speed tests and to keep the test costs under control, on-chip test solutions are needed. Such a solution is usually referred to as built-in self-test (BIST).

A BIST architecture consists of a test pattern generator (TPG), a test response analyzer (TRA) and a BIST control unit (BCU), all implemented on the chip. This approach supports at-speed tests and eliminates the need for an external tester. Different BIST approaches have been available for a while and have got wide acceptance especially for memory test. For logic BIST (LBIST) there is still no industry-wide acceptance. One of the main reasons is the hardware overhead required to implement a BIST architecture. The BIST approach can

also introduce additional delay to the circuitry and requires a relatively long test application time. Nevertheless, even LBIST is becoming increasingly popular, since BIST is basically the only practical solution to perform at-speed test, and can be used not only for manufacturing test but also for periodical field maintenance tests.

The classical way to implement the TPG for BIST is to use linear feedback shift registers (LFSR). However, the LFSR-based approach often does not guarantee a sufficiently high fault coverage (especially in the case of large and complex designs) and demands very long test application times in addition to high area overheads. Therefore, several proposals have been made to combine pseudorandom test patterns, generated by LFSRs, with deterministic patterns [1, 2, 12, 22, 24, 30, 33, 41, 48, 54, 57, 65, 66, 67, 76, 94, 95, 96, 97] to form a hybrid BIST solution.

The main concern of many existing hybrid BIST approaches has been to improve the fault coverage by mixing pseudorandom vectors with deterministic ones, while the issue of cost minimization has not been addressed directly.

In the following, two solutions are proposed to find the optimal balance between the on-line pseudorandom test pattern generation and usage of stored precomputed deterministic test patterns to perform core test with minimum cost of both time and memory, without losing test quality. Two algorithms will be described to calculate, with very low computational time, a complete hybrid test set, and to derive from it the optimal time-moment to stop pseudorandom test generation and to apply deterministic patterns.

A similar problem has been addressed in [57], where an approach to minimize testing time has been presented. The authors have shown that hybrid BIST (or Combination of BIST and External Test, CBET, in their terminology) can achieve shorter testing time than pure pseudorandom test or pure externally applied deterministic test. The authors have made a realistic assumption that externally applied test is much slower than LFSR generated one and therefore internally generated test vectors should be used as much as possible. However, the proposed algorithm is not addressing total cost minimization (time and memory) and is therefore only a special case of our approach.

4.2 Cost Factors for Hybrid BIST

As test patterns, generated by LFSRs, are pseudorandom by nature and have linear dependencies, the generated test sequences are usually very long and not sufficient to detect all the faults. To avoid the test quality loss due to random pattern resistant faults and in order to speed up the testing process, we have to apply deterministic test patterns targeting the random resistant and difficult to test faults. This hybrid BIST approach starts with on-line generation of pseudorandom test sequence with a length of *L*. On the next stage, stored test approach takes place. For the stored approach, precomputed test patterns, stored in the memory, are applied to the core under test to reach 100% fault coverage. For off-line generation of *S* deterministic test patterns (the number of stored test patterns) arbitrary software test generators may be used, based on deterministic, random or genetic algorithms.

In hybrid BIST, the length of the pseudorandom test is an important parameter, which determines the behavior of the whole test process. A shorter pseudorandom test set implies a larger deterministic test set. This however requires additional memory space, but at the same time, shortens the overall test process. A longer pseudorandom test, on the other hand, will lead to longer test application time with reduced memory requirements. Therefore it is crucial to determine the optimal length of pseudorandom test in order to minimize the total testing cost.

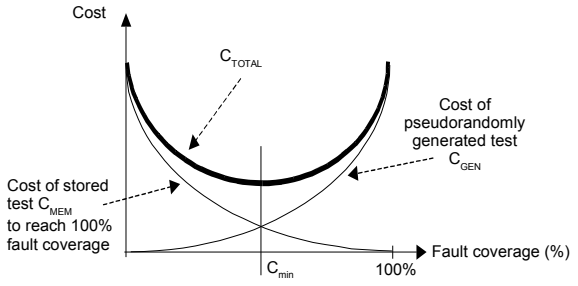


Fig. 4-1. Cost calculation for Hybrid BIST (presuming 100% coverage)

Fig. 4-1 illustrates graphically the total cost of a hybrid BIST consisting of pseudorandom test patterns and stored test patterns, generated off-line. A

situation is illustrated, where 100% fault coverage is achievable with pseudorandom vectors alone, although it takes enormously long time to do it. In the case of large and complex designs 100% fault coverage is not always achievable, however.

We can define the total test cost of the hybrid BIST C_{TOTAL} as:

$$C_{TOTAL} = C_{GEN} + C_{MEM}$$

where C_{GEN} is the cost related to the time for generating the pseudorandom test patterns (number of clock cycles), C_{MEM} is related to the memory cost for storing the precomputed test patterns to improve the pseudorandom test set.

Fig. 4-1 illustrates also how the cost of pseudorandom test increases when striving for higher fault coverage (the C_{GEN} curve). In general, it can be very expensive to achieve high fault coverage with pseudorandom test patterns only. The C_{MEM} curve describes the cost that we have to pay for storing additional precomputed tests from the fault coverage level reached by pseudorandom testing to 100%. The total cost C_{TOTAL} is the sum of the above two costs. The C_{TOTAL} curve is illustrated in Fig. 4-1, where the minimum point is marked as C_{min}. The main purpose of this work is to find a fast method for calculating the curve C_{TOTAL} to find the minimal cost C_{min}.

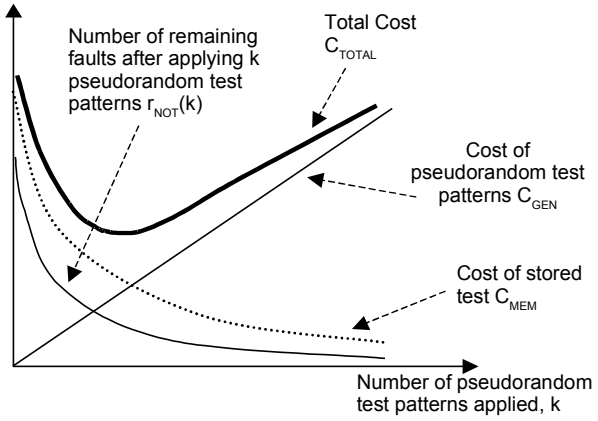


Fig. 4-2. Cost calculation for Hybrid BIST

As mentioned, in many cases 100% fault coverage is not achievable with pseudorandom vectors alone. Therefore we have to include this assumption to the total cost calculation and the new situation is illustrated in Fig. 4-2, where

the horizontal axis indicates the number of pseudorandom test patterns applied, instead of fault coverage level.

The curve for the total cost C_{TOTAL} is still the sum of two cost curves $C_{GEN}+C_{MEM}$ with the new assumption that the maximum fault coverage by using only deterministic ATPG is also achieved by the hybrid BIST.

We can also define the total cost of hybrid BIST C_{TOTAL} as:

$$C_{TOTAL}=\alpha L+\beta S$$

where L is the length of the pseudorandom test sequence; S is the number of deterministic patterns; and weights α and β reflect the correlation between the cost and the pseudorandom test time (number of clock cycles used) and between the cost and the memory size needed for storing the deterministic test sequence, respectively. For simplicity, we assume here that $\alpha=1$, and $\beta=B$ where B is the number of bytes of an input test vector to be applied to the CUT. Hence, in the following we will use, as the cost units, number of cycles used for pseudorandom test generation and number of bytes in the memory needed for storing the precomputed test patterns.

In Table 4-1, a fragment of the results of BIST simulation for the ISCAS'85 circuit c880 is given, where:

- k is the clock counter;
- $r_{DET}(k)$ is the number of new faults detected by the group of test patterns generated between the last and the current entry at k in the table;
- $r_{NOT}(k)$ is the number of faults not yet covered by the sequence of patterns generated during all the k clock cycles; and
- $FC(k)$ is the fault coverage reached by the sequence of patterns generated during all the k clock cycles.

Table 4-1. Pseudorandom test results for ISCAS'85 c880

k	$r_{DET}(k)$	$r_{NOT}(k)$	FC(k) %
1	155	839	15. 593561
2	76	763	23. 239437
3	65	698	29. 778671
4	90	608	38. 832996
5	44	564	43. 259556
6	39	525	47. 183098
10	104	421	57. 645874
16	66	355	64. 285713
20	44	311	68. 712273
29	42	269	72. 937622
50	51	218	78. 068413
70	57	161	83. 802818
100	16	145	85. 412476
149	13	132	86. 720322
200	18	114	88. 531189
323	13	101	89. 839035
412	31	70	92. 957748
708	24	46	95. 372231
955	18	28	97. 183098
1536	4	24	97. 585510
1561	8	16	98. 390343
2154	11	5	99. 496979
3450	2	3	99. 698189
4520	2	1	99. 899399
4521	1	0	100. 000000

In the list of BIST simulation results not all clock cycles are presented. We are only interested in the clock numbers at which at least one new fault will be covered, and thus the total fault coverage for the pseudorandom test sequence up to this clock number increases. Let us call such clock numbers and the corresponding pseudorandom test patterns resultative clocks and resultative patterns. The rows in Table 4-1 represent the resultative clocks, but not all (we only give some resultative points for illustrative purpose), for the circuit c880.

If we decide to switch from the on-line pseudorandom test generation mode to the deterministic stored pattern mode after the clock number *k*, then $L=k$.

Creating the curves C_{GEN} and $r_{NOT}(k)$ is not difficult. For this purpose, a simulation of the behavior of the LSFR used for pseudorandom test pattern generation is needed. A fault simulation should be carried out for the complete test sequence generated by the LFSR. As a result of such a simulation, we find for each clock cycle the list of faults which were covered up to this clock cycle. By removing these faults from the complete fault list, we will know the number of faults remaining to be tested.

More difficult is to find the values of βS , the cost for storing additional deterministic patterns in order to reach the given fault coverage level (100% in the ideal case). Let $t(k)$ be the number of test patterns needed to cover $r_{NOT}(k)$ not yet detected faults (these patterns should be precomputed and used as stored test patterns).

As an example, these data for the circuit c880 are depicted in Table 4-2. Calculation of the data in the column *t(k)* of Table 4-2 is the most expensive procedure.

Table 4-2. C880: ATPG results

k	*t(k)*	*k*	*t(k)*
0	104	148	46
1	104	200	41
2	100	322	35
3	101	411	26
4	99	707	17
5	99	954	12
10	95	1535	11
15	92	1560	7
20	87	2153	3
28	81	3449	2
50	74	4519	1
70	58	4520	0
100	52		

In the following, two algorithms for calculating *t(k)* will be compared: the algorithm presented in [23] and the algorithm developed in this thesis.

4.3 Fast Procedure for Calculating Stored Test Patterns

Calculation of the data in the column *t(k)* of Table 4-2 is the most expensive procedure in the hybrid BIST optimization process. In this section the difficulties and possible ways to solve the problem are discussed.

There are two possibilities to find t(k):

- ATPG based, and
- fault table based approach.

Let us have the following notations:

- i – the current number of the resultative clock cycle (the number of the entry in tables for PRG and ATPG
- $k(i)$ – the number of the clock cycle of the resultative clock i;
- $R_{DET}(i)$ - the set of new faults detected (covered) by the pseudorandom test pattern which is generated at the resultative clock signal number i;
- $R_{NOT}(i)$ - the set of not yet covered faults after applying the pseudorandom test pattern number i;
- $T(i)$ - the set of test patterns needed and found by the ATPG to cover the faults in $R_{NOT}(i)$;
- N – the number of all resultative patterns in the sequence created by the pseudorandom test;
- FT – the fault table for a given set of tests T and for the given set of faults R:

The fault table FT for a general case is defined as follows: given a set of test patterns $T=\{t_i\}$ and a set of faults $R=\{r_j\}, FT=\|\varepsilon_{ij}\|$ where $\varepsilon_{ij}=1$ if the test $t_i \in T$ detects the $r_j \in R$, and $\varepsilon_{ij}=0$ in the opposite case. We denote by $R(t_i) \subset R$ the subset of faults detected by the test pattern $t_i \in T$.

We start the procedure for a given circuit by generating a test set T which gives the 100% (or as high as possible) fault coverage. This test set can be served as a stored test if no on-line generated pseudorandom test sequence will be used. By fault simulation of the test set T for the given set of faults R of the circuit, we create the fault table FT.

Suppose now, that we use a pseudorandom test sequence T^L with a length L which detects a subset of faults $R^L \in R$. It is obvious that when switching from the pseudorandom test mode with a test set T^L to the precomputed stored test mode with a T, the test set T can be significantly reduced. At first, by the fault subtraction operation $R(t_i)-R^L$ we can update all the contributions of the test patterns t_i in FT (i.e. to calculate for all t_i the remaining faults they can detect after performing the pseudorandom test). After that we can use any procedure of static test compaction to minimize the test set T.

The described procedure of updating the fault table *FT* can be carried out iteratively for all possible breakpoints $i=1,2,\ldots,N$ of the pseudorandom test sequence by the following algorithms:

– The first algorithm was presented in [23] and represents the ATPG based approach:

Algorithm 4-1:

1. Take $i:=N$
2. Generate for $R_{NOT,i}$ a test set $T_i, T:=T_i, t_i:=|T_i|$
3. For all $i=N-1,N-2,\ldots,1$:
   Generate for the faults $R_{NOT,i}$ not covered by test *T*, a test set $T_i, T:=T+T_i,\ t_i:=|T|$

End.

– The second algorithm was developed in this thesis, and it represents the fault table approach:

Algorithm 4-2:

1. Calculate the whole test $T=\{t_j\}$ for the whole set of faults $R=\{r_j\}$ by any ATPG to reach as high fault coverage *C* as possible
2. Create for T and R the fault table $FT=\{R(t_j)\}$
3. Take $i=1$; Rename: $T_i=T, R_i=R, FT_i=FT$
4. Take $i=i+1$
5. Calculate by fault simulation the fault set $R_{DET,i}$
6. Update the fault table; $\forall j, t_j \in T_i: R(t_j)-R_{DET,i}$
7. Remove from the test set T_i all the test patterns $t_j \in T_i$ where $R(t_j)=\emptyset$
8. Optimize the test set T_i by any test compaction algorithm; fix the value of $S_i=|T_i|$ as the length of the stored test for $L=i$
9. If $i<L$, go to 4;

End.

It is easy to understand that for each value $L = i$ (the length of the pseudorandom test sequence) the procedure guarantees the constant fault coverage C of the hybrid BIST. The statement comes from the fact that the subset T_i of stored test patterns is complementing the pseudorandom test sequence for each $i = 1, 2, \ldots, N$ to reach the same fault coverage reached by T.

As the result of the algorithms, the numbers of precomputed deterministic test patterns $S_i = |T_i|$ to be stored and the subsets of these patterns T_i for each $i = 1, 2, \ldots, N$ are calculated. On the basis of this data the cost of stored test patterns for each i can be calculated by the formula $C_{MEM} = \beta S_i$. From the curve of the total cost $C_{TOTAL}(i) = \alpha L + \beta S$ the value of the minimum cost of the hybrid BIST $min\{C_{TOTAL}(i)\}$ can be easily found.

For the previously proposed Algorithm 4-1, the whole experiment of simulating the pseudorandom generation (PRG) behavior and of finding the numbers of test patterns to be stored for all possible switching points from PRG to stored test patterns was carried out for the whole set of ISCAS'85 benchmark circuits within about 8 hours. The data of these experiments are depicted in Table 4-3, page 42.

In the case of very large circuits both of these algorithms will lead to very expensive and time-consuming experiments. For such situations we have developed estimation algorithm to search for the optimum solution by using just a few samples from the whole test generation experiments set. As available data for such kind of estimation, the number of not yet covered faults in $R_{NOT,k}$ can be served. The value of $R_{NOT,k}$ can be acquired directly from the PRG simulation results and is available for every significant time moment (see Table 4-2). Based on the value of $|R_{NOT,k}|$ it is possible to reason about the expected number of test patterns needed for covering the faults in $R_{NOT,k}$. The starting point of the search procedure may be found by giving rough estimation of the total cost based on the value of $|R_{NOT,k}|$.

Experimental results of the comparison of the two described algorithms are presented in chapter 4.6.1.

4.4 The concept of the Method of Hybrid BIST with Reseeding

The idea of the method of hybrid BIST with reseeding, called also Store-and-Generate (S&G) method consists of using pseudorandom patterns generated by LFSR for detecting the random pattern testable faults, and the deterministic test patterns for testing random pattern resistant, so called hard-to-test-faults

(HTTF). To demonstrate this let us depict all possible 2^n-1 test patterns with size n as a line in Fig. 4-3. These patterns are in the order as they are generated by the LFSR. Let the dots below the line represent HTTF that can be tested by only a single test pattern. A pseudorandom test generated by the LFSR at the given seed (initial state) is shown as a small interval on the line (bold line). Here we see that many HTTF are outside of this interval and remain untested. Therefore we should construct a full test set by a collection of pseudorandom pattern blocks (PPB), represented in Fig. 4-4 as separate intervals, in such a way that all HTTF will be covered by these test patterns. Each block has its own seed. The main problem of this approach is how to calculate the number and size of PPBs, and how these tests should be spread over all test patterns, i.e. which should be the seeds for the PPBs. The limiting factor is that we don't know which faults are HTTF and which test patterns are needed for detecting them.

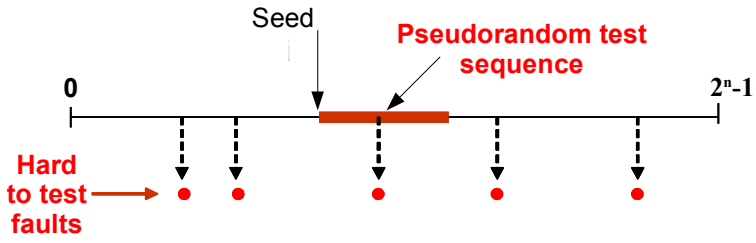


Fig. 4-3. Test patterns and hard-to-test-faults faults

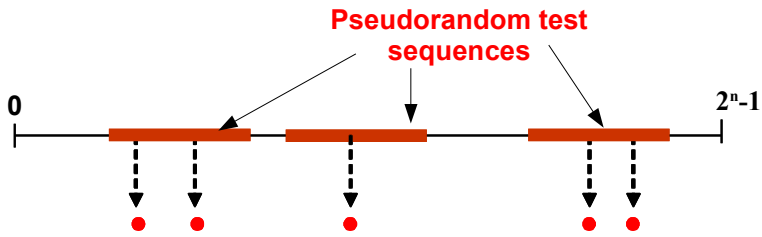


Fig. 4-4. Testing with many pseudorandom pattern blocks

Let *DT* be the deterministic test set for a given CUT, and *R* the set of all possible faults in the CUT. Let us call all the faults in $R_H \subset R$, which are covered only by a single test pattern in *DT*, HTTFs. With $DT_H \in DT$ we denote the subset of test patterns which cover HTTFs R_H. Obviously $|DT_H| = |R_H|$. The first seed $T_i \in DT_H$ for the first pseudorandom pattern block $B_i, i=1$, will be selected from the DT_H. Let $b_i = |B_i|$ be the length of the block B_i. The

algorithm now removes all the faults covered by B_i from R, and keeps in DT only these patterns that are needed for covering the faults in the updated R. A new $R_H \subset R$, and a new $DT_H \subset D$ are calculated, and the next seed $T_i \in DT_H, i=2$ from the updated DT_H will be chosen for starting the next block $B_i, i=2$, of pseudorandom patterns. This procedure should be continued till the set of faults R is empty. Let the number of iterations will be k. Then the length of the full test is calculated as

$$L = \sum_{i=i}^{k} b_i$$

The amount of memory M needed for storing the seeds is determined by k test patterns that are chosen for the k blocks. The characteristics of the solutions L and M are heavily depending on the length of the blocks. The task to be solved here is to find the lengths of the blocks so that $L = min$ at the given constraint $M \le M_{max}$.

We have carried out simulation as described in the procedure above for a range of different block lengths for the ISCAS benchmarks to see how the values of L and M are changing with the length of b. An example of the curves of $L_1(b)$, $L_2(b)$ and $M(b)$ for the ISCAS circuit c1908 is presented in Fig. 4-5, whereas for $L_1(b)$ the length of the blocks is equal, and for $L_2(b)$ the length of the blocks is different i.e. the block is cut shorter from the point where no useful patterns are generated.

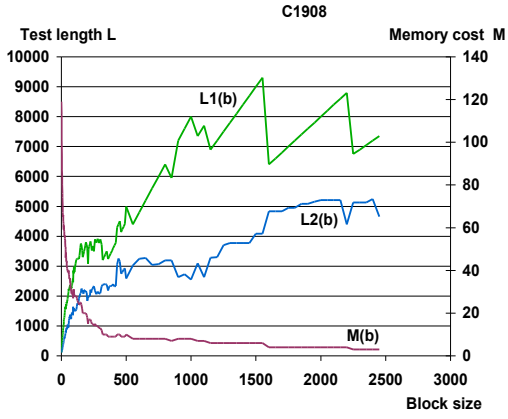


Fig. 4-5. Relationships L(b) and M(b) for c1908 circuit

In Fig. 4-6 a possible structure of the assumed BIST architecture is shown [107]. ROM contains the seeds. Each pattern P_i in the ROM serves as an initial state of the LFSR for test pattern generation. Counter 1 counts the number of L_i pseudorandom patterns that are generated starting from P_i. After finishing the cycle, Counter 2 is incremented for reading the next pattern P_{i+1}.

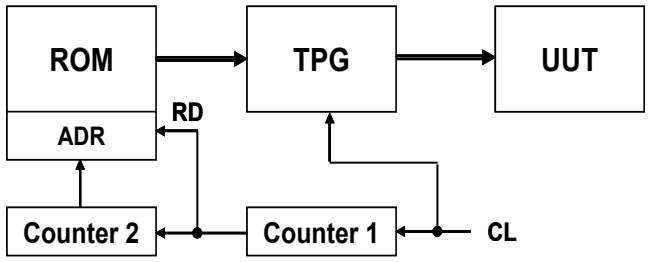


Fig. 4-6. Store and generate approach

4.5 Optimization of the Hybrid BIST with Reseeding

There are two important issues to consider when constructing reseeding based solutions: the number of seeds and the number of pseudorandom patterns generated from each seed. Both have significant impact on the final solution in terms of fault coverage, test length and test memory requirements. In the following a novel method is proposed for finding the length of the pseudorandom seeds, such that the test memory (number of seeds) is constrained, the test time is minimized, and the maximum achievable fault coverage is guaranteed. The pseudocode of the algorithm is depicted in Algorithm 4-3 and the details of the algorithm will be explained below.

```
Algorithm 4-3: pseudocode

- Generate deterministic test patterns DT, such that

- foreach DT_i:

        - Generate L pseudorandom patterns, using deterministic pattern as a seed;
        // *1

        - Fault simulate the block DT_i + PR_i

- block_length = 1;

- loop ( j <=L )

- foreach ( block ) // *2

-    calculate coverage summary ( block_length );

        // Find coverage summary for block with block_length number of vectors,
        // starting from the beginning of the block

-    minimize the block length;

        // find a point within a block from where the fault coverage does not
        increase                                                    anymore.
        // Removes the useless PR vectors

-    order blocks based on the length;

        // orders blocks in growing order (ie. shortest first)

-    minimize the number of blocks;

        //  finds  minimal  number  of  blocks  needed  to  obtain  the  maximum
        // fault coverage: remaining_blocks

-    initialize total_coverage_summary;

-    foreach( remaining_block );

        // calculate final test length, *3

-        optimize block length;

        //  remove  all  unnecessary  pseudorandom  patters  that  do  not  contribute
        // to the final fault coverage. Here we use accumulative coverage, ie.
        taking
        //  into  account  ALL  previously  applied  blocks.  For  this  we  use
        total_coverage_summary

-        total_coverage_summary += coverage( block );

-    end foreach;

        // add new faults, covered by this block

-    set new block_length;

        // block_length += block_length/stepping_const;

-    end loop;

- done;
```

*1 – see Fig. 4-7, step 1 ; *2 – see Fig. 4-7, step 2; *3 – see Fig. 4-8

Let DT be the deterministic test set $DT=\{DT_i\}$ for a given CUT and R the set of all possible faults in the CUT. Let us denote by $R(DT_i)\subset R$ the subset of faults detected by a test pattern $DT_i\in DT$. We assume that the DT obtains the maximum achievable fault coverage, hence $R(DT)=R$.

We start by generating pseudorandom sequences PR_i with a given length L, where DT_i is used as a seed for the pseudorandom sequence. Let us denote the set of these hybrid sequences as $PR=\{PR_i\}$. For all $PR_i\in PR$ and for each test pattern $t_k\in PR_i$, cumulative fault coverage can be calculated as:

$$FC(PR_{i,k})=\frac{\left|\bigcup_{j=1,k\,;\,t_j\in PR_i} R(t_j)\right|}{|R|} \qquad (4\text{-}1)$$

Thereafter the pseudorandom sequences can be minimized. From each $PR_i\in PR$ all pseudorandom test patterns $t_j,t_{j+1},\ldots,t_L$ where $FC(PR_{i,k})=FC(PR_{i,k-1})$, for all $k=j,j+1,\ldots,L$, will be removed, because these patterns will not contribute for the increase in fault coverage. These first steps are illustrated in Fig. 4-7. Let us denote with L_i the new reduced length of the pseudorandom sequence PR_i, with $FC(PR_i)$ the fault coverage of the sequence PR_i calculated using formula (4-1) of the last pattern in PR_i, and with $R(PR_i)\subset R$ the subset of faults detected by PR_i.

Thereafter, all hybrid test sequences PR_i will be ordered in increasing order, so that $L_i\geq L_{i-1}$ for every $i=1,2,\ldots,N$ where $N=|DT|$.

Consider now a composite hybrid test sequence PT, composed of all sequences $PR_i\in PR$ in the order how they were ranked. Since all initially generated deterministic test patterns $t\in DT$ are included in PT, we have

$$FC(PT)=FC(DT),$$
$$R(PT)=FC(DT)=R$$

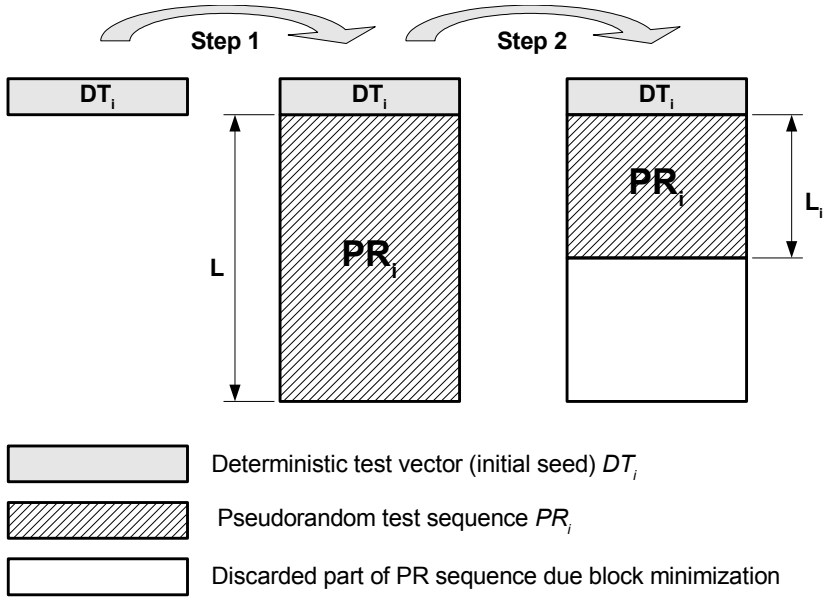


Fig. 4-7. Minimization of one pseudorandom sequence.

To minimize the length of the multi-seed hybrid test sequence we will use here the test pattern optimization algorithm(based on modified Greedy algorithm). In this algorithm the minimal subset T_{min} of the given test set T will be found based on the information of fault subsets $R(t)$ detected by the test patterns $t_i \in PR$, so that the fault coverage remains the same $FC(T_{min}) = FC(T)$. To do this we interpret the sequences $PR_i \in PR$ as test patterns $t \in T$, and the fault sets $R(PR_i)$, respectively as fault sets $R(t)$. As a result of optimization we will find a minimal subset $PR_{min} \subset PR$, so that $FC(PR_{min}) = FC(PR)$. This step is rather fast as we do not optimize at the level of individual patterns but only at the level of complete sequences (blocks).

Now a new composite hybrid sequence PT_{min} will be created from the ordered set of subsequences $PT_{min} = (PR_{1,} PR_{2,} \ldots, PR_m), m < N$.

The next step will be minimization of the total length of the sequence PT_{min}. As a result, a reduced final multi-seed hybrid sequence PT^* will be created.

To do that we calculate again the cumulative fault coverages for all the test patterns $t_k \in PT_{min}$ similarly to formula (4-1) for all subsequences $PR_i \in PR_{min}$ in the order how they were ranked and put into PT_{min}. After calculating the fault coverage of a current subsequence PR_i in PT^* we remove from PR_i all the test patterns $t_j, t_{j+1}, \ldots, t_{L,i}$ where

$FC(PR_{i,k}) = FC(PR_{i,k-1})$, for all $k = j, j+1, \ldots, L_i$. This procedure is illustrated in Fig. 4-8. As the optimization procedure takes into account only the cumulative fault coverage of earlier blocks and does not analyze individual patterns in these blocks, then also this step is rather fast.

Since the described reduction of the whole multi-seed hybrid sequence will not reduce the fault coverage, we have

$$FC(PT^*) = FC(PT) = FC(DT)$$

As a result of this algorithm we can find the length of a hybrid sequence for any arbitrary memory constraint. As a by-product we can also find the length of the longest hybrid block that remained at the end of the optimization sequence.

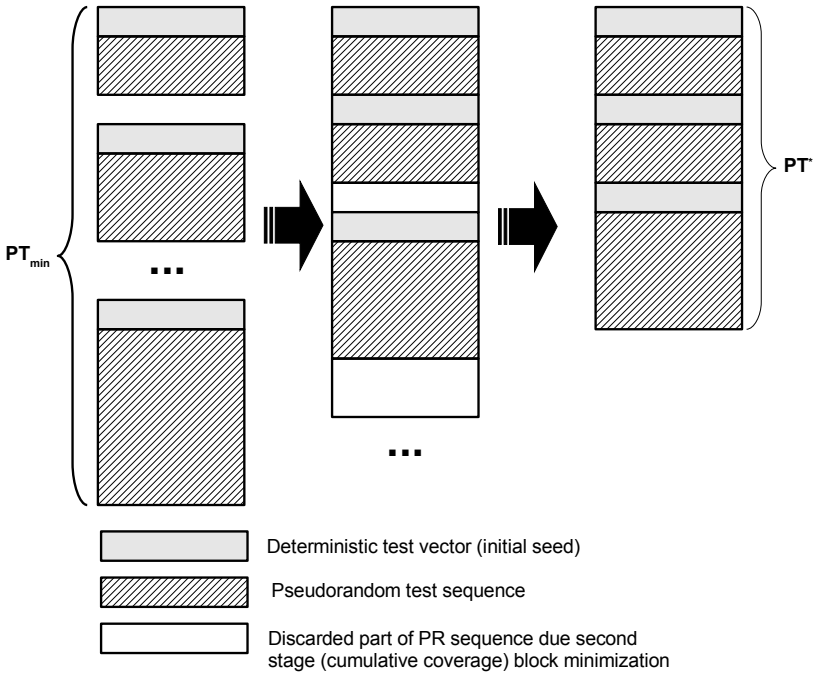


Fig. 4-8. Calculation of the final hybrid sequence.

Such an optimization is very necessary when developing a solution for testing core-based systems, such as SoCs or NoCs. The memory constraints can be seen as limitations of the on-chip memory or ATE, where the deterministic test set will be stored, and are therefore of great practical importance.

4.6 Experimental results

4.6.1 Optimization of the hybrid BIST

Experiments were carried out on the ISCAS'85 benchmark circuits for investigating the efficiency of the method for optimizing the hybrid BIST and for comparing with the previous algorithm used in [23]. Experiments were carried out using Turbo Tester [19, 119] toolset, for deterministic test pattern generation and fault simulation, and using the test compaction tool [3]. The results are presented in Table 4-3.

Table 4-3. Experimental results. Time comparison of two algorithms

Circuit	L_{PR}	L_{DET}	C_{PR}	C_{DET}	N	T_G	T_A	T_{OLD}	T_{NEW}	T_{OLD}/T_{NEW}
C432	780	80	93.02	93.02	81	20.10	0.01	1632.9	21.0	77.75
C499	2036	132	99.33	99.33	114	0.65	0.02	74.1	2.9	25.55
C880	5589	77	100.00	100.00	114	0.15	0.02	17.1	2.4	7.13
C1355	1522	126	99.51	99.51	109	1.22	0.03	133.0	4.5	29.56
C1908	5803	143	99.48	99.48	183	11.65	0.07	2132.0	24.5	87.02
C2670	6581	155	84.92	99.51	118	1.95	0.09	230.1	12.6	18.25
C3540	8734	211	95.54	95.54	265	85.29	0.14	22601.9	122.4	184.66
C5315	2318	171	98.89	98.89	252	10.29	0.11	2593.1	38.0	68.24
C6288	210	45	99.34	99.34	53	3.79	0.04	200.9	5.9	34.05
C7552	18704	267	93.67	97.14	279	53.78	0.27	15004.6	129.1	116.22

In the columns of Table 4-3 the following data is depicted: ISCAS'85 benchmark circuit name, L_{PR} - length of the pseudorandom test sequence, L_{DET} - the number of test patterns generated by the deterministic ATPG, C_{PR} - the fault coverage of the pseudorandom test sequence, C_{DET} - the total fault coverage of the hybrid BIST (after applying deterministic test patterns), N - number of all resultative patterns in the pseudorandom test sequence, T_G - the time (sec) needed for ATPG to generate the deterministic test set, T_A - the time (sec) needed for carrying out manipulations on fault tables (subtracting faults, and compacting the test set), T_{OLD} - the time (sec) needed for calculating the cost curve for hybrid BIST by the previous method , T_{NEW} - the time (sec) needed for calculating the cost curve for hybrid BIST by the method proposed in this thesis, the advantage of the proposed method compared to the previous one as the relation T_1/T_2 . The total testing time for both methods were calculated as follows:

$$T_{OLD} = N * T_G$$
$$T_{NEW} = T_G + N * T_A$$

In fact, the values for T_G and T_A differ for the different values of $i = 1, 2, \ldots, N$. However the differences were in the range of few percents, which allowed us to neglect this impact and to use the average values of T_G and T_A.

Table 4-4. Experimental results. Parameters for optimized hybrid BIST

Circuit	L_{MAX}	L_{OPT}	S_{MAX}	SOPT	B_k	CTOTAL
C432	780	91	80	21	4	186
C499	2036	78	132	60	6	386
C880	5589	121	77	48	8	481
C1355	1522	121	126	52	6	388
C1908	5803	105	143	123	5	612
C2670	6581	444	155	77	30	26867
C3540	8734	297	211	110	7	889
C5315	2318	711	171	12	23	985
C6288	210	20	45	20	4	100
C7552	18704	583	267	61	51	2161

The switching point from the PRG mode to the stored deterministic patterns mode was found at the minimum of C_{TOTAL}. In Table 4-4 the parameters of optimized hybrid BIST are depicted: ISCAS'85 benchmark circuit name, L_{MAX} - the maximum length of the simulated pseudorandom test sequence, L_{OPT} - the length of the pseudorandom test sequence for the optimized BIST, S_{MAX} - the maximum number of test patterns generated by the deterministic ATPG, S_{OPT} - the number of stored test patterns for the optimized BIST, B_k - the number of bytes needed for storing the input test pattern for the circuit k, and C_{TOTAL} - the total cost of the optimized hybrid BIST, calculated by the formula

$$C_{TOTAL} = L_{OPT} + B_k * S_{OPT}$$

In Fig. 4-9, the curves of the cost $C_{GEN} = L$ (denoted on Fig. 4-9 as T) for on-line pseudorandom test generation, the cost $C_{MEM} = B_k * S$ (denoted as M) for storing the test patterns, the number $|R_{NOT}|$ of not detected faults after applying the pseudorandom test sequence (denoted as F_r), and the total cost function C_{TOTAL} are depicted for selected benchmark circuits C432, C499, C880, C1908, C3540 and C7552 (Sc = 0 is used as a constant in the cost function formula).

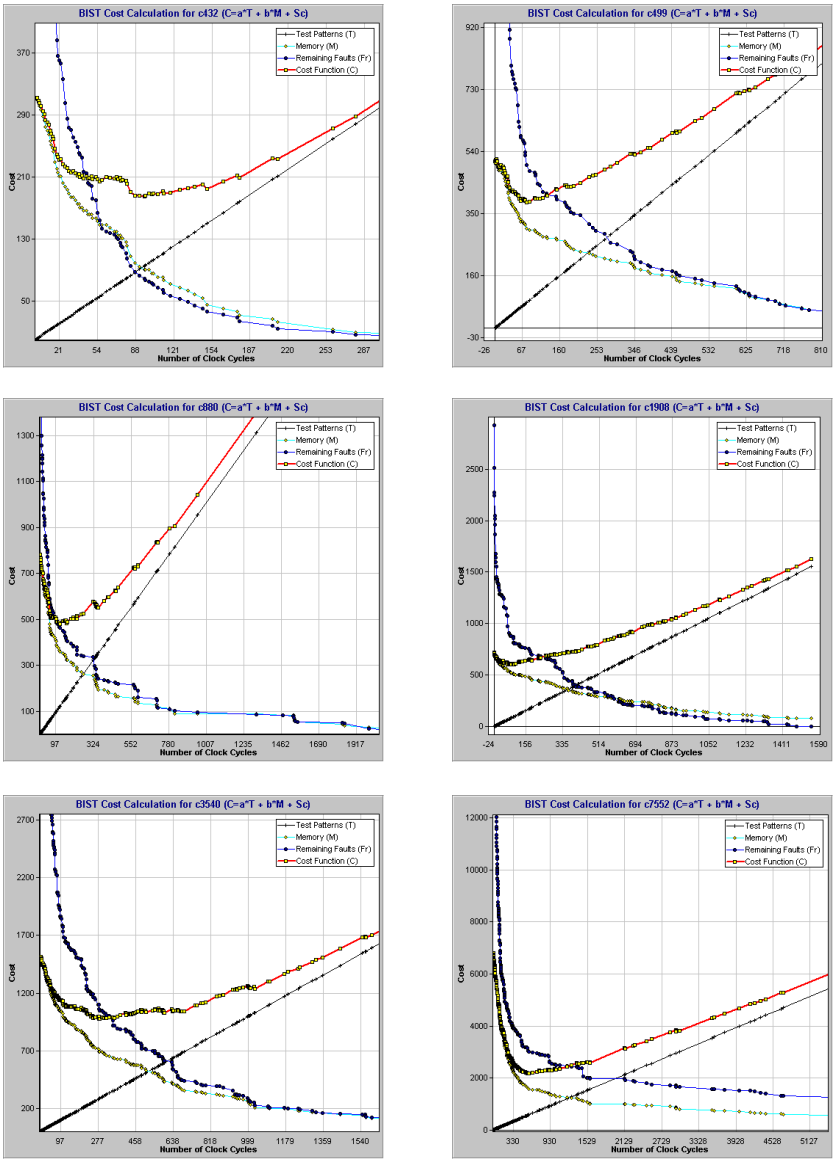


Fig. 4-9. Cost curves of hybrid for ISCAS'85 benchmark circuits

4.6.2 Optimization of the hybrid BIST with reseeding

Experiments were carried out on the ISCAS'85 benchmark circuits for investigating the efficiency of the method for reducing the test length based on the proposed algorithm of calculating the hybrid test sets for each possible memory constraint. For some tasks (like ATPG and fault simulation) tools from the Turbo Tester toolset [119] were used.

Table 4-5. Experimental results

Circuit	Memory Constraint (vectors)	Proposed method		Hybrid BIST *1	Reseeding *2
		Max block size	Total length	Total length	Total length
C432	10	11	96	206	280
	15	7	81	188	210
	20	4	60	151	160
	25	3	62	120	125
	30	2	51	79	120
C499	20	23	318	492	940
	30	10	211	326	540
	40	6	153	193	400
	50	4	125	149	350
	60	3	105	115	240
C1908	30	30	777	1318	1680
	40	21	617	869	1240
	50	13	444	735	1050
	58	10	337	667	870
	61	9	303	600	793
C1355	15	37	412	438	615
	30	10	195	282	930
	41	6	156	207	492
	49	4	120	182	343
	52	3	109	179	312
C2670	50	15	272	598	6400
	54	8	211	342	2538
	56	7	171	311	1568
	60	4	140	290	1080
	71	2	104	150	923
C3450	60	6	258	414	900
	70	5	224	293	700
	80	3	167	253	560
	90	2	142	201	360
C5315	15	40	541	753	1290
	35	10	237	451	560
	42	5	173	299	504
	46	4	134	268	368
	51	3	124	199	255
C7552	95	10	414	900	1140
	102	6	333	500	714
	115	3	210	334	460
	124	2	165	192	372

1 - described in [22]; 2* - described in [88]*

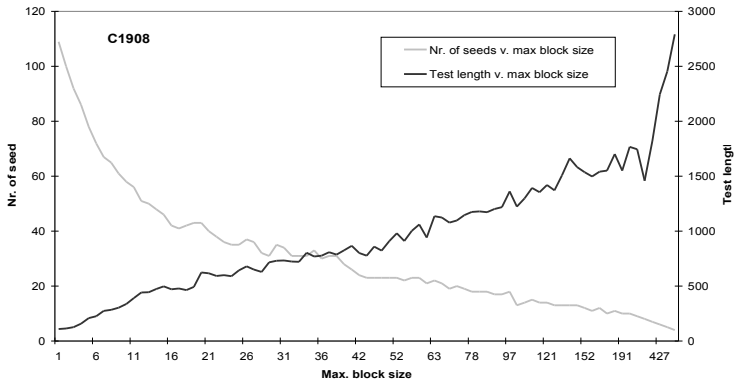
The results are presented in Table 4-5, where we have depicted the test length of the final solution under different memory constraints. We have compared the method, proposed in this paper, with methods proposed in [22] and [83]. The method proposed in [22] was originally developed for multi-core systems but it can equally well be used also for individual cores. In addition we have depicted in Table 4-5 also the lengths of the longest hybrid block at the end of the optimization cycle.

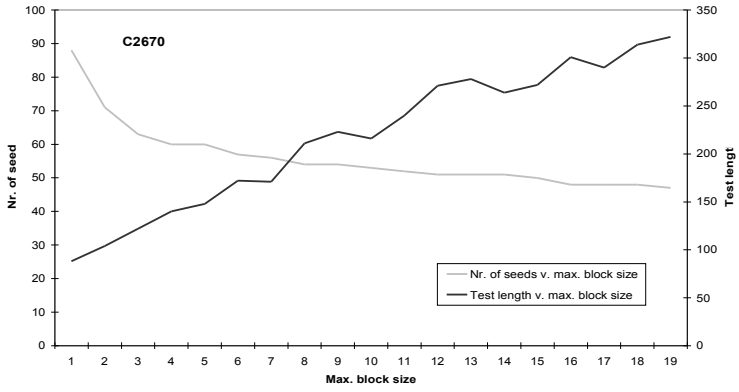
As it can be seen from the results then the proposed method can always find a test set that is shorter than the test set found using methods from [22] and [83]. The main explanation lies in a fact that the proposed method handles the deterministic and pseudorandom sequences together and the test sets are optimized using a fast optimization method, based on cumulative fault coverage figures. The current implementation of the method [83] is not optimizing the length of the individual hybrid blocks and therefore also the result worse than the result from [22].

Although the experiments depicted here, were performed on combinatorial circuits, the proposed method can with some small modifications also handle the sequential circuits with full scan (using appropriate test architecture, such as STUMPS).

In Fig. 4-10 and Fig. 4-11 we have depicted more detailed results of some selected circuits.

In Fig. 4-10 we have illustrated the relationships in between memory constraint (number of seeds), test length and the length of the longest block. As it can be seen from these charts then reduction of the number of seeds will increase the length of the hybrid blocks and consequently also the test length will be increased.





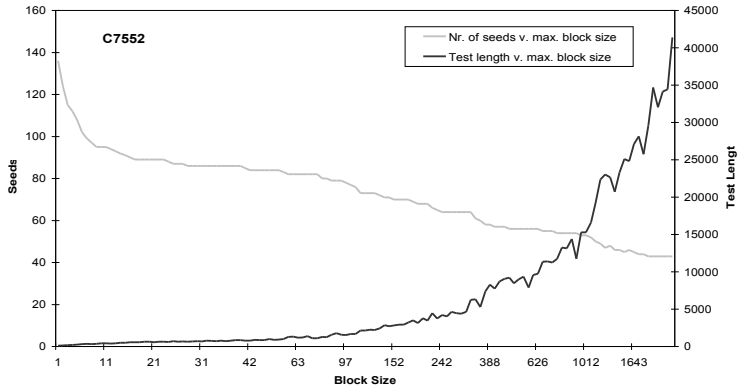


Fig. 4-10. Relationships in between number of seeds, maximal block size and total test length.

In Fig. 4-11 this relationship is illustrated in more straightforward manner, showing how the test length is increasing while the number of seeds is reducing.

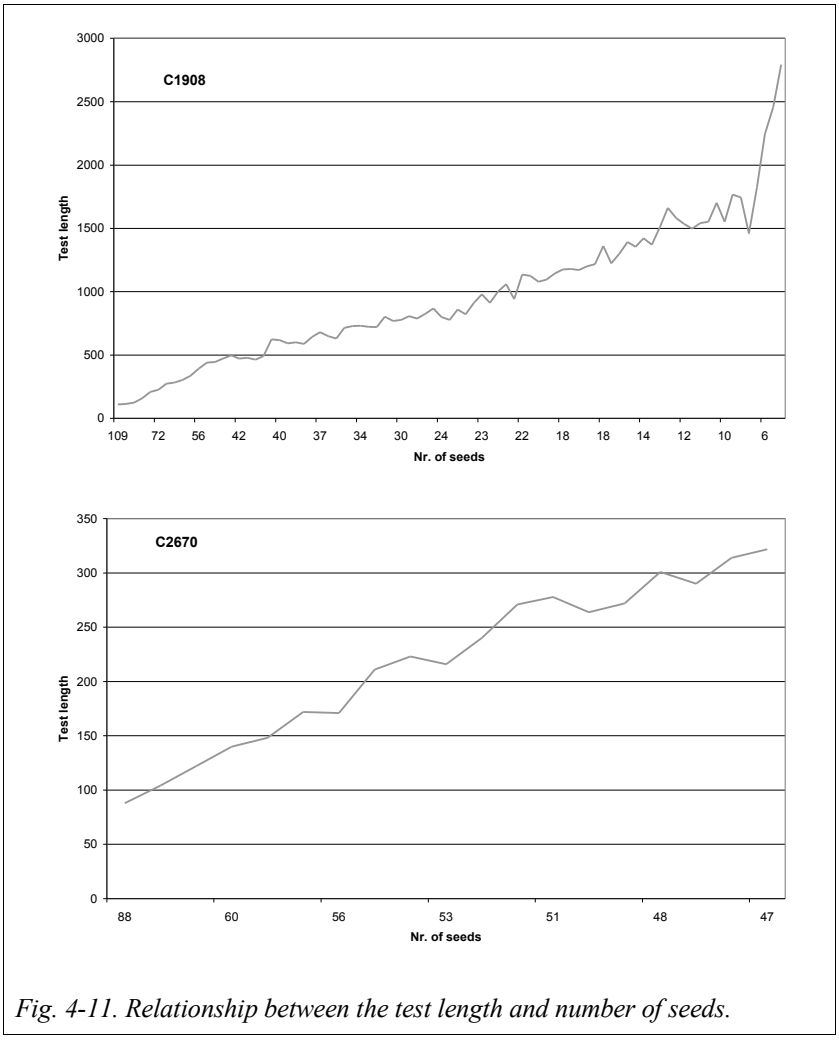


Fig. 4-11. Relationship between the test length and number of seeds.

We have not included the CPU times here, as with the proposed method we are calculating the entire solution space, while the methods from [8] and [98] can find a single solution for a predefined memory constraint. We are currently developing an optimization heuristic that would help us to avoid calculation of the complete curves and thereafter the comparison of CPU times is also possible.

4.7 Conclusions

1. A hybrid BIST solution has been developed for testing systems-on-chip. It combines on-line pseudorandom test pattern generation with using precomputed and stored deterministic test patterns.

2. For selecting the optimal switching moment from pseudorandom patterns mode to the stored deterministic patterns mode, a fast algorithm to calculate the total cost was developed.

3. It was shown by experimental results, that the new cost calculation algorithm calculates the total cost of the hybrid BIST much faster (7 - 184 times faster) than previously proposed algorithm and can therefore speed up the total cost minimization process significantly.

4. A method is developed for optimization of the hybrid BIST, based on reseeding. An optimization heuristic is proposed for test length reduction under given memory constraints, based on the test set compaction using cumulative fault coverage of hybrid test sequences.

5. Experimental results have shown that the proposed approach is feasible and efficient for finding optimized solutions for hybrid BIST architectures, based on the reseeding concept.

5 Hybrid Functional Built-In Self-Test

A method for designing hybrid functional BIST (HyBIST) is proposed to combine the functional routines carried out in digital systems with deterministic test patterns for testing microprogrammed data-paths in digital systems. In the first test phase only the functional resources of a system are used for testing purposes. A functional microprogram is carried out to control the data-path based on some deterministic input data. A response compressor like signature analyzer is connected to the data path to monitor the process.
To guarantee a high test coverage for BIST, the second phase of the test is used which consists of applying additional deterministic test patterns pre-generated by an ATPG to test the random-pattern-resistant faults. A method is proposed to find the trade-off between the functional test and deterministic test parts. Experiments demonstrate the feasibility of the approach, and also show the advantage of combining functional and deterministic test patterns compared to the pure deterministic test or pure functional test.

The results of this chapter are published in [86].

5.1 Principles of Hybrid Functional BIST

Rapid advances in the areas of deep-submicron electron technology and design automation tools are enabling engineers to design larger and more complex circuits and to integrate them into one single chip. System on a Chip (SoC) design methodology is seen as a major new technology and the future direction for semiconductor industry. The most important challenges of SoC testing are linked to test cost and fault coverage. According to the ITRS (International Technology Roadmap for Semiconductors) by 2014 it may cost more to test a transistor than to manufacture it unless techniques like logic Built-in Self-Test BIST are employed [51]. BIST is a technology to move on board the main functionalities previously carried out by Automated Test Equipments (ATE). In traditional BIST architectures, test pattern generation is mostly performed by *ad hoc* circuitry, typically Linear Feedback Shift Registers (LFSR) [51], cellular automata [71] or multifunctional registers like BILBO (Built-in Logic Block Observer) [16]. BIST involves using on-chip hardware to apply pseudorandom test patterns to the Circuit Under Test (CUT) and to analyze its output response. The most widespread approach is test-per-scan BIST scheme [16].

Unfortunately, many circuits contain random-pattern-resistant faults [67] which limit the fault coverage that can be achieved with this approach.

One method for improving the fault coverage for a test-per-scan BIST is to modify the CUT by either inserting test points [55, 64] or by redesigning it to improve the fault coverage [18, 114]. The drawback of these techniques is that they generally add additional logic levels to the circuitry that can degrade system performance. Fault coverage can be improved by another way by using weighted pseudorandom sequences. Additional logic is needed to weight the probability of each bit in the test sequence. The weight logic can be placed either at the input of the scan chain [61] or in the individual scan cells themselves [8, 96]. The disadvantage of the probability weighting approach is in the need of storing of the weight sets on chip and also, control logic is required to switch between weights, so the hardware overhead may be large.

A third method to improve the fault coverage is to use a “mixed mode” approach where deterministic patterns are used to detect the faults that the pseudorandom patterns miss. Storing deterministic patterns may require a large amount of hardware overhead. In [8] a technique based on reseeding an LFSR was proposed that reduces the storage requirements. In [96] another improved technique was developed that uses a multi-polynomial LFSR for encoding a set of deterministic test cubes. Many other improvements of BIST have been discussed in the previous chapters

Established BIST solutions use special hardware for pattern generation (TPG) and test response evaluation (TRE) on chip, but this in general introduces significant area overhead and performance degradation. To overcome these problems, recently new methods have been proposed which exploit specific functional units such as arithmetic units or processor cores for on-chip test pattern generation and test response evaluation [40, 75, 76, 92, 95]. In particular, it has been shown that adders can be used as TPGs for pseudo-random, pseudo-exhaustive and deterministic patterns. Investigations are known about properties of test patterns generated by simple adders [40], ones- and twos complemented subtractors [7], and more complex multipliers and MAC circuits [39]. All of them may generate pseudo-exhaustive or pseudorandom patterns with a similar quality as LFSRs do, and may reach a comparable fault coverage.

The term "functional BIST" (FBIST) describes a test method to control functional modules so that they generate a deterministic test set, which targets structural faults within other parts of the system. It is a promising solution for self-testing complex digital systems at reduced costs in terms of area overhead and performance degradation.

In this work, a mixed-mode or hybrid functional BIST (HyFBIST) was developed for using in microprogrammed data-paths in digital systems. The idea of the HyFBIST consists in using for test purposes the mixture of functional patterns produced by the microprogram, and additional stored deterministic test patterns to improve the total fault coverage.

In the first phase a microprogram (as a part of the functionality of the system) is used to control the data-path based on some deterministic or random input data. A response compressor like signature analyzer is connected to the data path to monitor the process. The data produced by the microprogram are used for both, stimulating the units under test and creating the signature of the process. The second phase of the test consists of applying additional deterministic test patterns pregenerated by an ATPG to test the random-pattern-resistant faults, which are stored in the memory. A method is proposed to find the tradeoff between the functional test and deterministic test parts.

5.2 General Scheme of Hybrid Functional BIST

Consider a microprogrammed data-path for division of fractional numbers, presented in Fig. 5-1. It consists of a register block for storing the dividend, the divisor, intermediate results of division, the quotient, and the counter of cycles. All the microoperations needed in the division procedure are carried out in the Arithmetic and Logic Unit (ALU) which has the role of CUT in this work. The ALU has data inputs and outputs connected via buses to the register block. The control signals from the control unit serve as additional inputs for ALU, and status signals of the ALU serve as additional outputs connected to the control unit (not shown in Fig. 5-1).

During N cycles of the microprogram ALU is exercised with N functional patterns, and the responses of ALU will be compressed in the signature analyzer which monitors the whole division process.

In the division process, we could use just K pairs of the operands A and B involved as the test for the ALU, and K quotients $C = A/B$ as K responses to the test stimuli. However, in the FBIST scheme we will use all the $K*N$ data words produced on the inputs of the ALU during the $K*N$ cycles of the K division operations as input stimuli to the ALU, and all the $K*N$ data produced on the outputs of the ALU during the $K*N$ cycles as the responses to stimuli. In such a way, we have got a multiplication effect of N times in the

number of test patterns when moving the test access from the instruction level to the microinstruction level.

Denote by L the number of bits in the data (dividend and divisor), and by l the number of bits on the inputs of ALU. The reduction in the test data volume through the compression of test data in the FBIST is equal to

$$R = \frac{N\,l}{2\mathrm{L}}$$

For example (for the system used in the experiments), in the case of 32 bit words for the divisor with 105 inputs and 120 cycles the reduction in the volume of test data is 120*105/64 = 197.

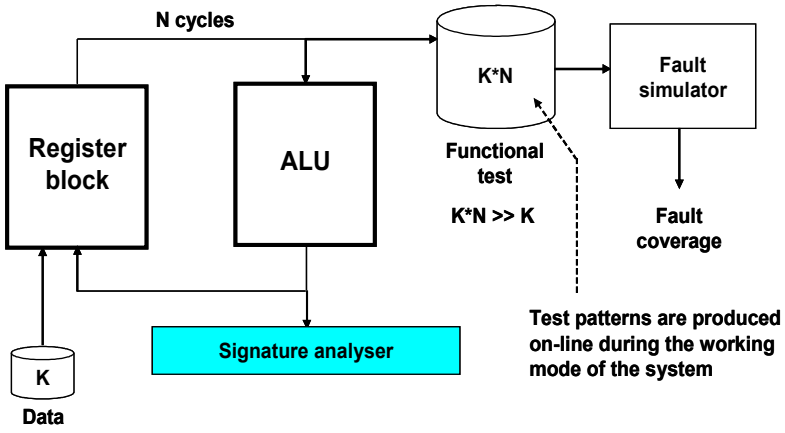


Fig. 5-1. Functional BIST quality analysis in the microprogrammed divisor

In this scheme the functional patterns produced directly on the inputs of ALU have the similar role as pseudorandom test patterns in classical BIST schemes. Similarly to the pseudorandom test, the functional test patterns are not able to cover random-pattern-resistant faults, which limits the fault coverage that can be achieved with the pure functional BIST approach.

To improve the fault coverage we can use similar approaches that are used to improve the LFSR-based classical BIST approaches: to modify the CUT by inserting test points, by redesigning it to improve the fault coverage, or by using hybrid approaches, adding to functional test additional deterministic test patterns.

In Fig. 5-1 the quality of the set of functional test patterns generated during the division procedure will be measured by fault simulation, the random-pattern-resistant faults are determined, and to cover these faults, additional deterministic test patterns by an ATPG are generated.

Such a hybrid functional test is carried out in two phases (Fig. 5-2). In the first phase the microprogram (as a part of the functionality of the system) is used to control the data-path based on some deterministic or random input data (operands). A response compressor like signature analyzer is connected to the data path to monitor the process. The data produced by the microprogram are used for both, stimulating the CUT and creating the signature of the process.

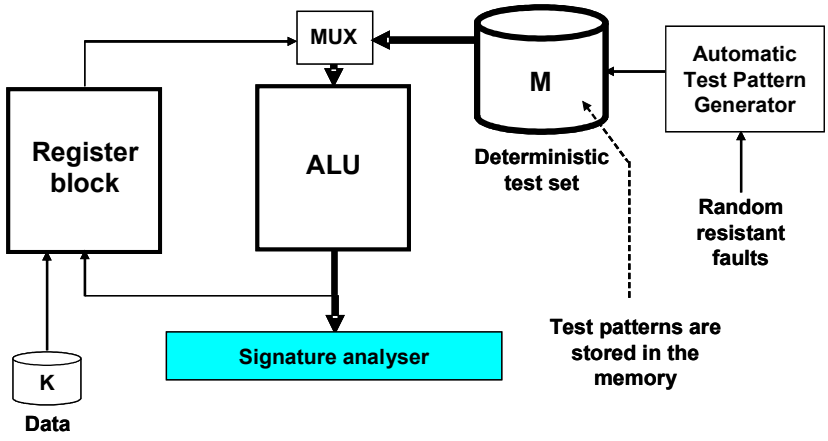


Fig. 5-2. Functional BIST with adding deterministic test patterns

The second phase of the test consists of applying additional deterministic test patterns pre-generated by an ATPG to test the random-pattern-resistant faults, which are stored in the memory.

Further, a method is proposed to find the trade-off between the functional test and deterministic test parts.

5.3 Finding Trade-off Between Functional and Deterministic Test patterns

This hybrid FBIST approach starts with on-line generation of functional test sequence with a length of $2kL$ where L is the length of the data word in bits, and k is the number of data operands used for producing the functional test sequence. $2kL$ is the memory cost for the functional part of the test. On the next phase, deterministic test approach takes place. Precomputed deterministic test patterns, stored in the memory, are applied to the CUT to reach 100% fault coverage. For the off-line generation of D deterministic test patterns (D is the number of test patterns to be stored), arbitrary software test generators may be used, based on deterministic, random or genetic algorithms.

The length of the functional test (the number of data operands) is an important parameter, which determines the structure and the quality of the whole test process. A shorter functional test set implies a larger deterministic test set. This however requires additional memory space, but at the same time, shortens the overall test process. A longer functional test, on the other hand, will lead to longer test application time, however, with reduced memory requirements, since the functional test data is more tightly compressed. Therefore it is crucial to determine the optimal length of functional part of the test in order to minimize the total test cost.

Consider the total test cost C_{TOTAL} of the hybrid FBIST as the sum of total costs C_{FB_Total} and C_{D_Total}, correspondingly, of producing functional and deterministic test patterns

$$C_{Total} = C_{FB\_Total} + C_{D\_Total}$$

where

$$C_{FB\_Total} = C_{FB\_Const} + \alpha C_{FB\_T} + \beta C_{FB\_M} \text{ , and}$$

$$C_{D\_Total} = C_{D\_Const} + \alpha C_{D\_T} + \beta C_{D\_M}$$

Here C_{FB_Const} (C_{D_Const}), C_{FB_T} (C_{D_T}), and C_{FB_M} (C_{D_M}) mean, correspondingly, additional logic cost, the cost related to the time used for testing, and the cost of additional memory needed for functional and deterministic test parts, whereas α and β reflect the weights, of time and memory expenses. An example of the cost curves is shown in Fig. 5-3.

Creating the curve of C_{FB_Total} is not difficult. The static component C_{FB_Const} is related to the cost of signature analyzer, and the dynamic components are determined linearly by the number of test operands used for the functional test whereas

$$C_{FB\_T} = \alpha \sum_{j=1}^{k} N_j$$

is the number of clocks (time cost) used for carrying out the microprogram, and

$$C_{FB\_M} = \beta 2\text{kL}$$

is the number of bits (memory cost) needed for storing the data operands.
For simplicity we take $\alpha = 1$, and $\beta = 1$. Hence, in the following we calculate the time cost by the number of clocks used for carrying out the test, and the memory cost by the number of bits needed for storing the precomputed test data.

The static component C_{D_Const} of the deterministic test is related to the cost of multiplexer on the inputs of ALU and to the cost of an additional microprogram needed for carrying out the deterministic part of the test.

For calculating the dynamic part of the the cost of deterministic test,

$$C_{D\_T} = \alpha D \quad \text{and} \quad C_{D\_M} = \beta Dl \ ,$$

the not tested by FBIST faults are found by fault simulator, and the number of additional patterns D of the deterministic test is calculated by Algorithm 5-1.

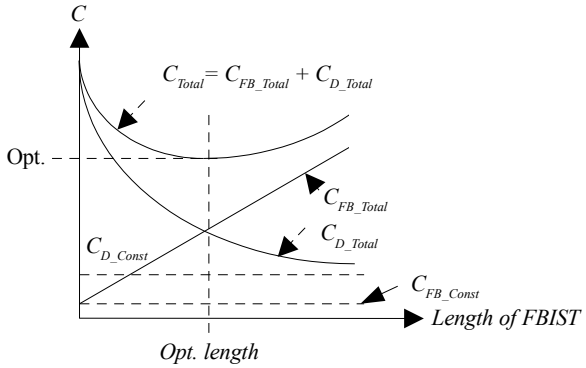


Fig. 5-3. Cost curves for HyFBIST

Algorithm 5-1:

1. Take $j = 0$; calculate the whole deterministic test $T_D(j)$ for the whole set of faults $R(j)$ in the CUT.
2. Create the fault table $FT(j)$
3. FOR all $j = 1, 2, \ldots, k$:
   BEGIN
   - Find the first pair of data operands (A_j, B_j);
   - Carry out the functional test with (A_j, B_j) and find the set of N_j functional test patterns;
   - Fault simulate the N_j patterns produced by the functional test, and find the set of faults $R_{DET}(j)$ detected;
   - Create a new fault table $FT(j)$ by removing from $FT(j-1)$ the faults $R_{DET}(j)$ and optimize the deterministic test $T_D(j-1)$ in relation to $FT(j)$;
   - The optimized new test set is $T_D(j)$ with the length $D(j) = |T_D(j)|$;

   END FOR

END.

Using the values $D(j)$ found by Algorithm 5-1 for each possible length $j = 1, 2, \ldots, k$ of the functional test, it is possible to create the curve of the cost C_{D_Total} of the deterministic test, and the curve of the total cost C_{Total} of HyBIST. By finding the minimum of C_{Total} we can determine the optimal mixture of the functional and deterministic parts of HyBIST.

5.4 Experimental results

Experiments were carried out for the microprogrammed data path for division of fractional numbers presented in Fig. 5-4.

The data path has 105 inputs, and 71 outputs, it consists of three 32- bit registers (dividend, divisor and quotient), 5-bit counter, and a combinational part of 513 gates. The fault list of the UUT consists of 2382 faults.

A series of experiments was carried out to determine the fault coverage which may be achieved by a single division microprogram. The results are depicted in Table 5-1 where *A* is the dividend, *B* is the divisor, *C* is the quotient, *N* is the number of cycles carried out during the microprogram, and *FC* is the fault coverage reached by the N functional test patterns produced by the microprogram on the inputs of ALU. For this UUT a single microprogram as a functional test allows test data compression in 197 times (see Section 5.2).

From Table 5-1 we see that a single division procedure for a single pair of data operands A and B is not able to produce a high fault coverage by using the proposed functional BIST scheme.

The second series of experiments was carried out to merge several division procedures into a single functional test program.

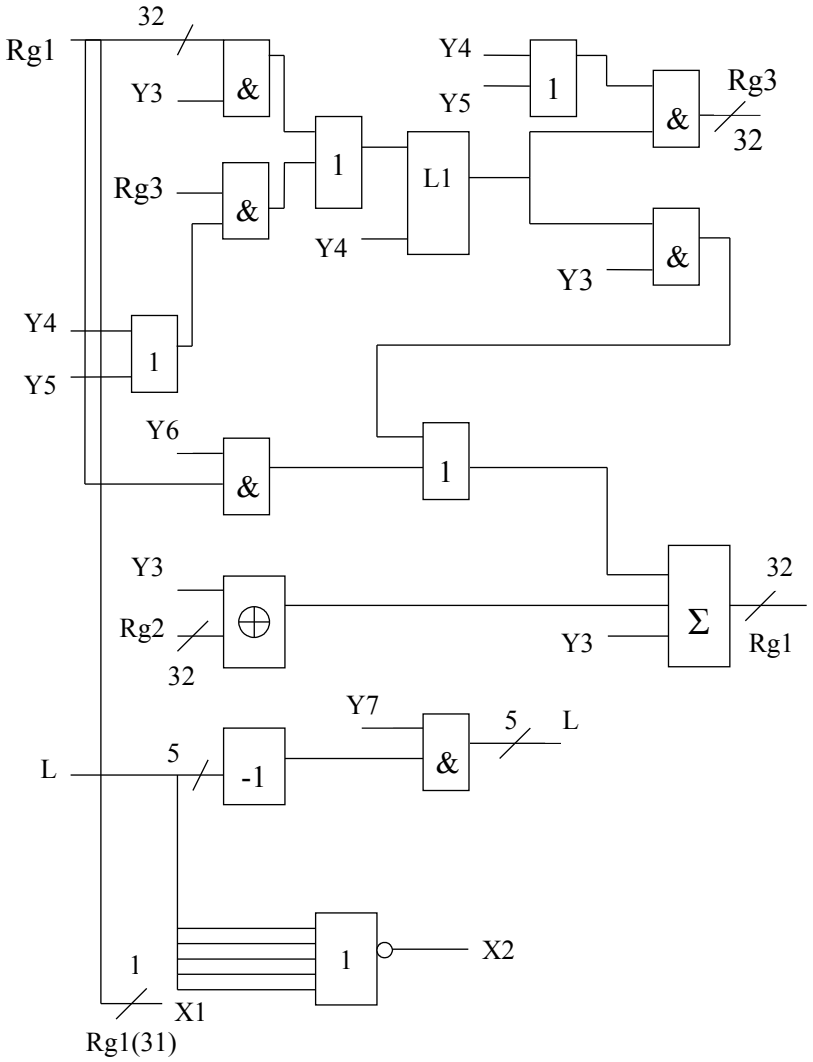


Fig. 5-4. Unit under test used in the experiments

The sequences of up to 10 division microprograms with different data pairs (*A,B*) were carried out to calculate for each case the optimum combination of functional and deterministic test parts. A selection of 6 experiments is presented in Table 5-2. Here *k* is the optimal number of runs of the microprogram (optimal length of the functional test part), *N* is the number of functional test patterns produced by k microprograms, *FC* is fault coverage, and *D* is the number of additional deterministic test patterns generated to achieve 100% fault coverage

for the whole hybrid FBIST procedure. The total costs are calculated for both, functional and deterministic test parts, and for the whole hybrid FBIST. For simplicity we have taken a = b = 1, and $C_{FB\_Const} = C_{D\_Const} = 0$. The best results of each column are marked by bold.

Table 5-1. Selected functional tests implemented as a single division procedure

No	*A*	*B*	*C*	*N*	*FC(%)*
1	0.5000	0.5000	1.0000	94	42.48
2	0.2500	0.5000	0.5000	124	44.87
3	0.1500	0.1500	1.0000	94	48.78
4	0.4000	0.8000	0.5000	124	52.64
5	0.2000	0.8000	0.2500	124	56.38
6	0.5000	0.8000	0.6250	99	64.48
7	0.9043	0.9865	0.9167	108	65.07
8	0.2953	0.3456	0.8545	109	66.20
9	0.6943	0.7234	0.9598	105	66.96
10	0.4320	0.8569	0.5041	113	67.25
11	0.4567	0.4678	0.9763	104	67.51
12	0.4320	0.5678	0.7608	108	67.84
13	0.4320	0.6000	0.7200	108	68.01
14	0.7435	0.8764	0.8484	104	68.30
15	0.4320	0.4509	0.9581	107	68.89

Table 5-2. Selected optimal test procedures

Functional test part				Deterministic test part		Total cost
k	*N*	*FC %*	Total cost	*D*	Total cost	
4	430	89.1	686	16	1696	2382
3	329	84.7	521	16	1696	2217
4	438	83.7	694	16	1696	2390
3	293	69.0	485	22	2332	2817
3	282	69.1	474	22	2332	2806
2	213	76.7	277	18	1908	2185

From Table 5-2 we see that the minimal total cost essentially depends on the data operands chosen for the functional test. The task of generating the best data operands for the functional test part was not the task in this work. The goal in the

thesis was to minimize the whole cost of the hybrid FBIST at the given functional test.

In Table 5-3, the progress of parameters at the increasing length k of the functional test part is shown for the best combination of functional and deterministic test parts. The cost curves for this experiment are shown also in Fig. 5-5 with minimum total cost at $k=2$. For this solution, the functional part of the hybrid BIST consists of two runs of the microprogram for two pairs of stored 32-bit data operands, and of the deterministic part with 18 stored 105-bit test patterns. The cost of optimized HyBIST C_{Total_opt} = 2185 compared to the pure deterministic testing $C_{D\_Total\_pure}$ = 6148 is 2.8 times less that characterizes the gain achieved by the presented approach..

Table 5-3. Calculation of data for optimization

Functional test part					Deterministic test part		Total cost
k	N_j	*N*	*FC %*	Total cost	*D*	Total cost	
0	0	0	100	0	58	6148	6148
1	108	108	66.8	140	24	2544	2684
2	105	213	76.7	277	18	1908	2185
3	113	326	83.3	518	17	1802	2320
4	108	434	85.5	690	16	1696	2386
5	110	544	88.4	864	15	1590	2454

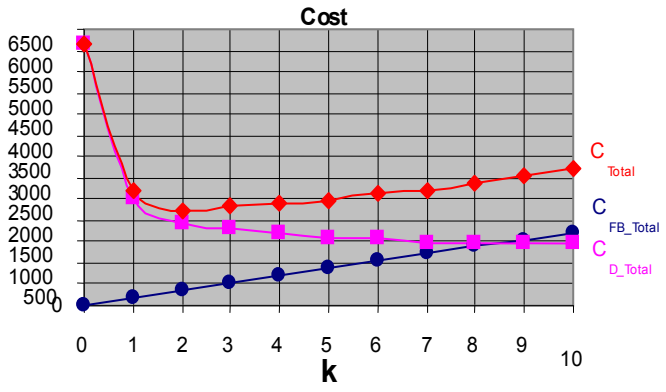


Fig. 5-5. Cost curves for the HyFBIST

Table 5-4. Comparison of functional test with FBIST

Data	Functional testing			Functional BIST		
	B1	B2	Total	B1	B2	Total
4/2	13.21	15.09	14.15	35.14	40.57	29.72
7/2	21.23	16.98	19.10	38.44	47.64	29.25
6/3	19.34	31.6	25.47	41.04	39.62	42.45
8/2	25.47	10.38	17.92	32.07	40.57	25.00
9/4	8.96	5.66	7.31	36.56	47.64	25.47
9/3	**32.55**	26.89	29.72	43.63	46.07	40.57
12/6	13.44	8.02	18.87	36.08	39.62	32.55
14/2	18.16	25.00	11.32	37.50	49.06	25.94
15/3	29.48	**31.13**	**27.83**	**47.88**	**50.00**	**45.75**
2/4	7.8	7.55	8.02	29.01	20.75	33.02
Average	18.96	17.83	17.97	37.74	42.15	32.97
Gain	1.0	1.0	1.0	2.0	2.4	1.8

A research environment for investigating different aspects of RT level design and test was recently developed [93] where the described functional BIST approach can be compared with traditional functional RT level testing. There is a possibility in the applet to run different microprograms (taken from the library or user defined) for arbitrary chosen data operands, and calculate the sensitivity of the final results of microprograms to all possible stuck-at faults in the data path. This corresponds to traditional functional testing where a computer instruction is exercised with different data. Another mode of working of the applet is functional BIST where signature analyzers can be inserted at different places of the circuit to increase the observability of the microprogram behavior in its role as a test.

The results of comparison of the traditional functional testing and functional BIST for the integer division microprogram by using the applet [93] are given in Table 5-4.

Two blocks B1 and B2 were selected for comparison. The gains in fault coverage in favour for FBIST are 2.0, 2.4 and 1.8 times in average, correspondingly, for selected blocks B1, B2 and in total for the whole data path.

The fault coverage reached by FBIST was higher in the experiments presented in Table 5-1 compared to the results in Table 5-4. This can be explained by the difference in the lengths of data words. In the educational applet [93] only 4-bit

data are used (for making processes visually better observable) whereas the results in Table 5-1 are based on 32-bit data.

5.5 Conclusions

1. A new approach to functional BIST was developed which uses three ideas:
   - functional data generated cyclically online in a digital system are used as test patterns,
   - these functional data are combined with additional deterministic test patterns to achieve 100% fault coverage, and
   - an optimal combination of both test parts is found.

2. Experiments showed the feasibility and efficiency of this approach. The data compression in FBIST was 197, and the gain in test cost for HyBIST compared to pure deterministic testing was 2.8 times.

3. As we see from the experimental results, the global problem of finding hybrid FBIST with minimum cost is more complex than only finding the best functional part, and then optimizing the HyFBIST. The experiments showed that the best functional test part with highest fault coverage (89,1% in Table 5-2) does not guarantee the best HyFBIST. These investigations targeted to optimization of HyFBIST in a global sense need future research..

6 Sequential Built-In Self-Test

In this chapter a Design-for-Testability (DfT) technique of Built-In Self-Test (BIST) for sequential circuits is proposed. The technique is based on making the status signals entering the control part controllable during the test mode to force the device under test to traverse all the branches in the FSM state transition graph. Extra outputs are added to the circuit under test in order to observe the values of the status bits masked out.

The proposed idea of architecture requires little device area overhead since a simple controller can be implemented to manipulate the control signals. Experiments were carried out on six sequential examples in order to compare different approaches to sequential BIST.

6.1 Principles of Sequential Built-In Self-Test

Here we propose a technique known from software testing to be implemented in Built-In Self-Test (BIST) for synchronous sequential circuits. Here, path coverage metrics s[8] is used to generate masks for controlling the FSM of the device under test in the test mode. In addition, general architecture to allow such application of the masks is proposed. It contains an LFSR and a simple controller to manipulate the masked out bits. Due to the small number of such signals in most of the circuits, very little area overhead is required.

The problem of Built-In Self-Test (BIST) for combinational and full-scan circuits has been thoroughly researched in the past. Implementing weighted random test patterns [18, 31] and bit-flipping [33] have been among the most efficient solutions.

Nachman, Saluja et al. [49] propose an a method were the values are held at inputs and scan registers while a certain number of clock pulses are applied. This requires preliminary testability analysis of the circuit structure. Furthermore, the above approach is applicable for circuits containing scan-chains only.

However, rather limited amount of work is available on BIST for non-scan sequential designs. The main motivation for sequential BIST is that, unlike in scan-chain approach, there is no need to reconfigure the circuit flip-flops during the test mode. This allows testing of the circuit at its normal operating speed.

Pomeranz and Reddy present a solution for the general case of sequential circuits [36]. However, the main problem is an excessive hardware overhead since dedicated test pattern generators are to be tailored for each individual primary input. In [46], Chakrabarty proposes a method similar to the reseeding approach [97]. Here, deterministic patterns are embedded to a sequence generated on-chip by using twisted ring counters (or Johnson counters, as they are also referred to).

The Chapter is organized as follows. Section 2 explains the functional fault model of covering all the branches in an FSM state transition graph. Section 3 presents the proposed DfT-based BIST approach technique. In Section 4, experimental results are provided. Finally, main conclusions are given.

6.2 Test Coverage Metrics for Sequential Circuits

Consider a Finite State Machine (FSM) as a model for sequential circuits. FSM may be represented using a state transition table or a state diagram. A state diagram is a directed graph, where the nodes correspond to states and the branches correspond to transitions between the states. Marked on the branches are the conditions required to activate them. In a digital system, these conditions usually correspond to status bits originating from conditional operations in the datapath.

The approach proposed here is based on all-branches coverage metrics [56], which is known to be more powerful than all-statement coverage. Let us consider an example in Fig. 6-1, where covering all the branches in the state transition graph of the FSM is presented.

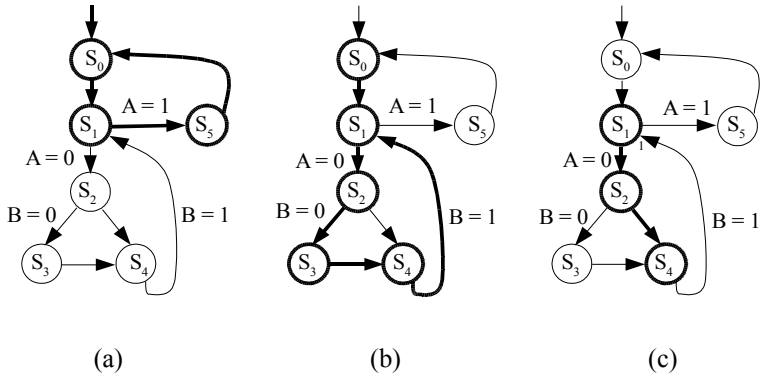


Fig. 6-1. Traversing all branches in the state transition graph

In Fig. 6-1a, we traverse a sequence s0 → s1 → s5 → s0 by setting the status signal A to be 1. Fig. 6-1b shows traversing the next sequence s0 → s1 → s2 → s3 → s4 → s1 by assigning $A:=0$ and $B:=0$. Finally, the sequence s1 → s2 → s4 is covered by assigning 1 to the status bit B (Fig. 6-1c). As it can be seen, all the branches of the state transition graph for the example FSM are covered by the paths in Fig. 6-1.

The main idea of current approach is to force the FSM to traverse all the branches in the state transition graph. This is implemented by controlling the status bits entering the control part and feeding pseudorandom data to the primary inputs of the circuit. The next Section explains this architecture more in detail.

6.3 General Architecture of the BIST

Fig. 6-2 presents the general architecture of the DfT enhanced BIST. It contains a Pseudo Random Pattern Generator (PRPG), a BIST controller, Circuit Under Test (CUT) and a MUX to select between the normal inputs and the pseudorandom test. Linear Feedback Shift-Register (LFSR) has been implemented as the PRPG.

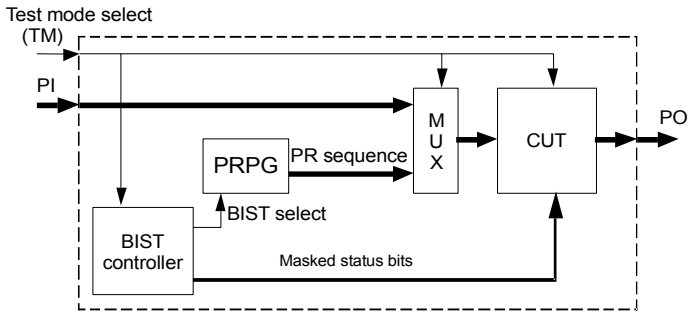


Fig. 6-2. General BIST architecture for status bit masking

The task of the BIST controller is to activate the pseudo-random pattern generator and control the values of the status bits. The pseudorandom test generation in the experiments were carried out by CAD tools belonging to Turbo-Tester [58]. Output response (signature, aliasing) analysis was not considered here.

Fig. 6-3 shows the structure of a digital system modified according to the DfT approach. The circuit under test is divided into an FSM and a datapath. The DfT architecture implements multiplexers to mask out the status signals of the datapath entering the FSM. Normal status bit values are selected during the working mode (TM=0) and controller-generated masked values during the test mode (TM=1). The muxed-out signals are made observable by adding dedicated observation points.

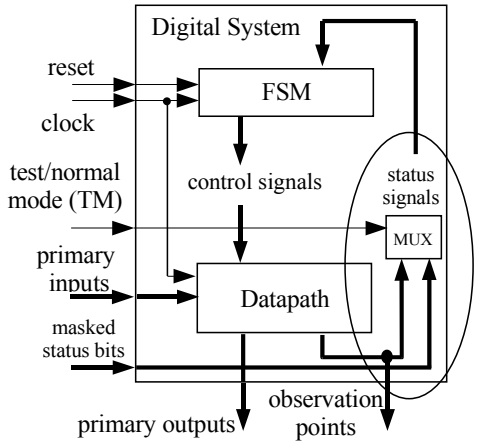


Fig. 6-3. Digital system modified for DfT

As we can see in the experiments presented in the following Section, the number of such signals is usually very low (from zero to two in the considered benchmarks). Thus the area overhead required by the controller and the MUX-es is low.

6.4 Experimental Results

Table 6-1 presents characteristics of the benchmark circuits that have been chosen from the HLSynth92 [118] and VILAB benchmark families [117]. The circuits with '_MOD' extension are the modified designs, where the test mode multiplexer has been inserted and the status bits have been made controllable.

Table 6-1. Characteristics of the benchmark circuits

Circuit	FSM states	PI bits	PO bits	registers	MUXes	FUs	faults	mask bits	area overhead
DIFFEQ	6	81	48	7	9	5	10360	-	-
DIFFEQ_MOD	6	83	48	7	10	5	10372	1	0.1 %
ELLIPF	28	130	113	17	7	3	5674	-	-
GCD	8	9	4	3	4	3	452	-	-
GCD_MOD	8	12	4	3	6	3	474	2	4.4 %
MULT8x8	8	17	16	7	4	9	2064	-	-
MULT8x8_MOD	8	20	16	7	6	9	2098	2	1.7 %
RISC	4	26	16	8	4	4	6418	-	-
SOSQ	5	9	32	7	2	6	1952	-	-
SOSQ_MOD	5	11	32	7	3	6	1964	1	0.5 %

The last column of the Table 6-1 shows the area overhead imposed by the status-bit multiplexers. As we can see, the number of mask bits and therefore, the number of additional multiplexers, is very low. Thus, the required overhead of the multiplexers is neglectable ranging from 0.1 % to 4.4 %.

Table 6-2 shows the average and maximum fault coverages for all the benchmarks both, for 1000 and 10000 pseudorandom vectors. Five different test configurations were considered:

- LFSR and original circuit.
  The original circuit was tested with pseudorandom patterns generated by an LFSR.
- LFSR and modified circuit (_MOD)
  The modified circuit (i.e. the circuit, where status bits have been made controllable) was tested with an LFSR.
- LFSR and test masks in test mode (TM=1)
  The modified circuit was tested with an LFSR, the test masks were applied in the test mode (TM signal was active).
- LFSR and test masks in normal mode (TM=0)
  The modified circuit was tested with an LFSR, but the test masks were applied in the normal working mode (TM signal was deactivated).
- LFSR with reset handling (Reset)
  The modified circuit was tested with an LFSR and the global reset was kept deactivated during each test sequence.

Table 6-2. Comparison of sequential BIST solutions

Circuit	average coverage %		maximal coverage %	
	1000	10000	1000	10000
MULT8x8	1.55	1.55	1.55	1.55
MULT8x8_MOD	5.90	5.90	5.90	5.90
MULT8x8_MOD (TM=1)	47.63	47.98	47.95	48.05
MULT8x8_MOD (TM=0)	5.67	5.67	5.67	5.67
MULT8x8_MOD (reset)	**49.71**	**58.29**	**54.10**	**58.52**
SOSQ	3.63	3.27	3.63	3.63
SOSQ_MOD	10.49	9.62	10.58	10.58
SOSQ_MOD (TM=1)	**46.63**	**47.71**	**47.71**	**47.71**
SOSQ_MOD (TM=0)	3.60	3.60	3.61	3.61
SOSQ_MOD (reset)	38.49	36.70	42.22	42.27
ELLIPF	4.96	5.26	5.59	5.80
ELLIPF (reset)	**85.00**	**85.01**	**85.02**	**85.02**
DIFFEQ	93.03	94.27	93.35	94.55
DIFFEQ_MOD	94.15	95.53	94.56	**95.96**
DIFFEQ_MOD (TM=1)	94.63	95.39	**95.06**	95.88
DIFFEQ_MOD (TM=0)	**94.75**	**95.40**	94.92	95.74
DIFFEQ_MOD (reset)	94.63	95.32	94.78	95.44
GCD	39.56	50.44	53.08	68.72
GCD_MOD	56.13	71.26	79.41	84.66
GCD_MOD (TM=1)	**85.95**	85.95	86.13	86.13
GCD_MOD (TM=0)	81.76	84.73	84.87	84.87
GCD_MOD (reset)	84.39	87.29	**88.03**	**88.03**
RISC	29.00	**40.03**	33.05	42.43
RISC (reset)	**36.87**	39.50	37.91	39.55

Fig. 6-4 gives a clearer view of the results presenting the performance of the above-mentioned techniques on the six circuits. The data is presented for the maximal results obtained with 1000 clock-cycles. We can distinguish between several types of circuits with different characteristics. There are two circuits, which are well random-testable: GCD and DIFFEQ. As it can be seen from the Fig. 6-4, all the bars for these circuits are of nearly similar height and reach nearly 100 %. This means that for these circuits any BIST scheme will do, including the pure pseudo-random approach. Another circuit that can be easily tested by pseudorandom data is RISC. However, here the main reason is most likely the very small sequential depth (4 clock-cycles).

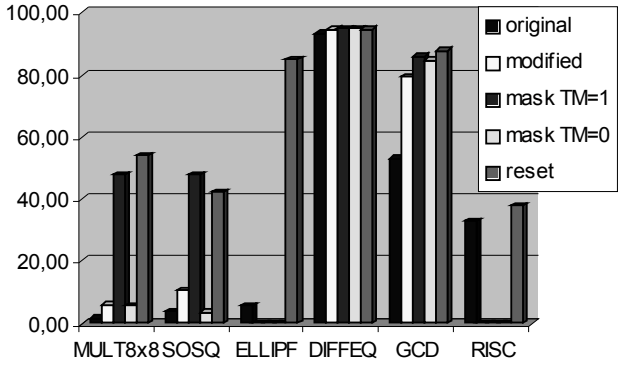


Fig. 6-4. Effect of DfT on different example circuits

The rest of the circuits can not be efficiently tested by pseudo-random vectors. While ELLIPF and MULT8x8 could be well tested by simple reset handling, for the SOSQ benchmark, signal masking should be preferred. Thus, depending on the pseudo-random testability characteristics, an appropriate approach can be selected for each individual case.

6.5 Conclusions

1. A new technique known from software testing was introduced to be implemented in Built-In Self-Test for synchronous sequential circuits. Path coverage metrics was used to generate masks for controlling the FSM of the device under test. In addition, general architecture to allow such application of the masks was proposed.

2. Experiments carried out on six sequential benchmarks showed that most of the circuits could not be tested by pseudorandom data. Controlling the FSM of the circuit under test considerably improved the results.

3. The experiments also showed that there was no universally better solution among the compared architectures. Depending on the pseudo-random testability characteristics, an appropriate approach has to be selected for each individual test case.

7 Environment for e-Learning in Digital Test

A set of tools developed as prototypes for experimental research of new BIST approaches described above has been accommodated and extended for using in education purposes.

In this Chapter these tools (as "interactive modules") are presented and it is shown how they can be used as e-Learning environment for teaching and learning logic level test generation and fault diagnosis problems in digital circuits. The tools can support laboratory work for different university courses on computer engineering, switching and automata theories, digital electronics and design for testability to learn by hands-on exercises 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.

The results in this chapter are published in [77, 78, 79, 80, 81, 89]

7.1 Applet "Introduction to Digital Test"

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 [63]. This reflects also today's university education: everyone learns about design, but only truly dedicated students learn testing. 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 Section 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 [85]. 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 [85]. 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.

In the core of the teaching concept presented here 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 behavior of the circuit in the fault-free mode and also in different faulty modes. In the following paragraphs, different learning tasks and exercises are described which make use of this applet.

7.1.1 User Interface

The program for representing “living pictures” for teaching Digital Test was originally written in Java 1.3. It can be run over network, using standard browsers like Netscape and Internet Explorer with Java 1.3 (or newer) runtime plug-in, or with Java 1.3 (or newer) 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. 7-1):

- 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 [51]. 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” [107], or other hybrid BIST approaches discussed on this thesis.

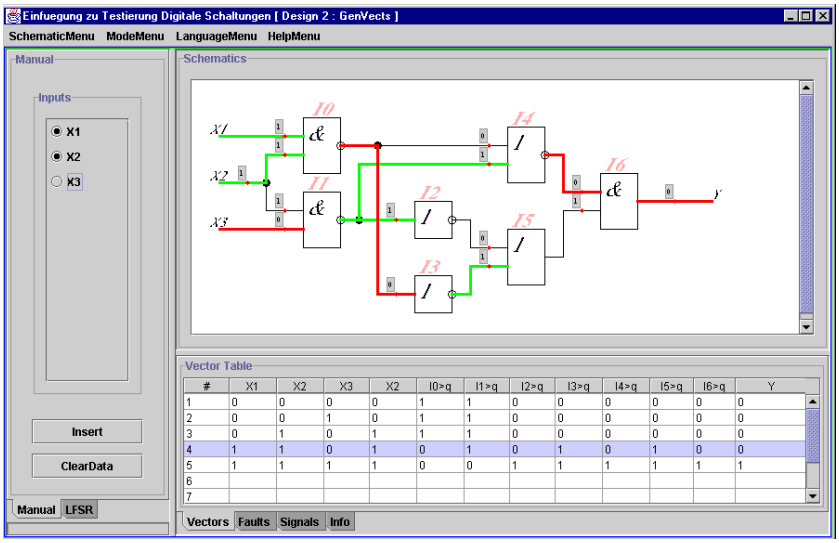


Fig. 7-1. Working window of the applet

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 encoding is as follows:

- 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 it is possible to 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 it is possible to 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 important 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.

7.1.2 Test Vector Generation

There are two methods possible for test vector generation using 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).

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) [51]. 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 circuit 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 colors 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 color. 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.

For example, to generate a test pattern for the fault $x_3 \equiv 1$ in Fig. 7-2, 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*.

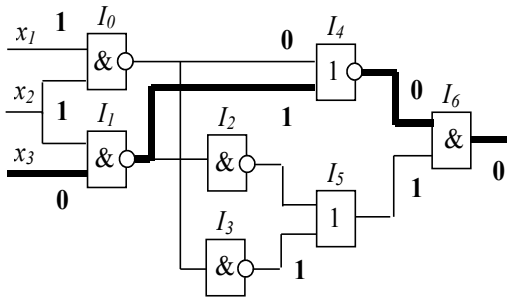


Fig. 7-2. Test generation by path activation

The activated path is shown in Fig. 7-2 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 $x_1 x_2 x_3 = 110$ 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 colors on the schematic panel.

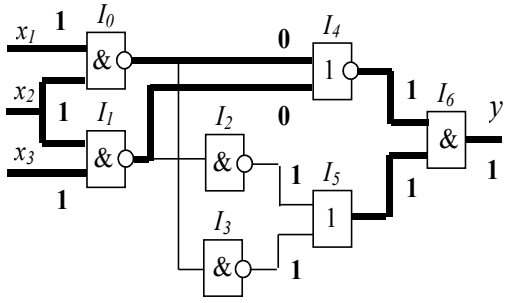


Fig. 7-3. Fault simulation results

For example, in Fig. 7-3, activated paths (shown by bold lines) are found by fault simulating the test pattern $x_1 x_2 x_3 = 111$.

The following faults are detected along these paths:

$x_1 \equiv 0, x_2 \equiv 0, x_3 \equiv 0$

$I_0 b \equiv 0, I_1 a \equiv 0$

$I_4 a \equiv 1, I_4 b \equiv 1$

$I_6 a \equiv 0, I_6 b \equiv 0$

$y \equiv 0$

The inputs of gates are denoted from above down by a, b.

In Fig. 7-1, 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 behavior of connection lines of the circuit for this test vector. The activated faults are highlighted by colored lines, the value 0 (or 1) in the boxes means that the fault stuck-at-1 (or stuck-at-0) is activated.

7.1.3 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 lies in backtracing 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 $x_1 x_2 x_3 = 110$ represented in Fig. 7-2. Suppose the gate I_1 is faulty. An error has been observed on the output. Three possible fault location procedures can be imagined.

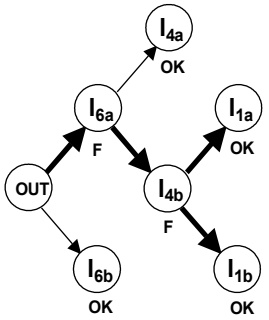


Fig. 7-4. Fault diagnosis by backtracing errors

First, we may use a trivial backtracing procedure of erroneous signals shown as a search tree in Fig. 7-4. 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_4a. 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 analyze 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. 7-4).

There is a third possibility to analyze the situation for a global optimization of the search 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.

7.1.4 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 [51] consisting of the following main components: test pattern generator (TPG), unit under test (UUT) and a response analyzer (RA). The corresponding BIST architecture is shown in Fig. 7-5.

TPG is usually a pseudorandom test pattern generator, and RA – a signature analyzer, 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. 7-5. BIST architecture (BILBO)

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 pre-generated test patterns

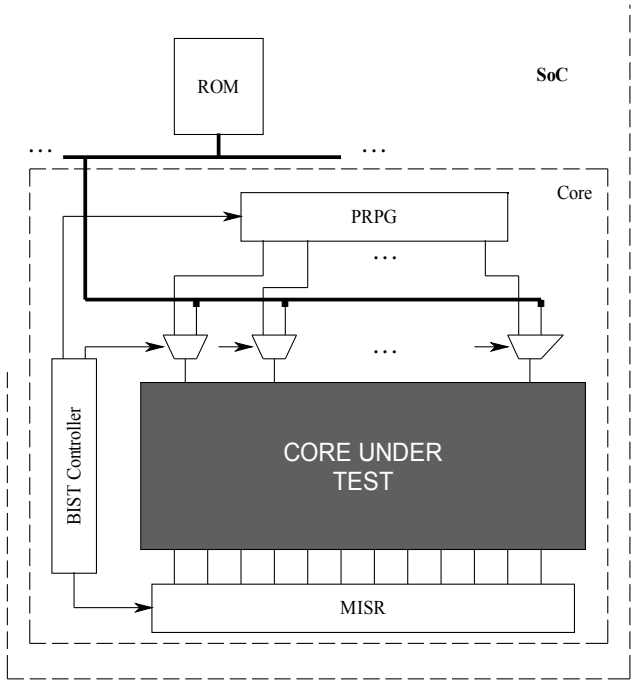


Fig. 7-6. BIST architecture (BILBO)

A hybrid BIST architecture is depicted in Fig. 7-6. 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 trade-off 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.

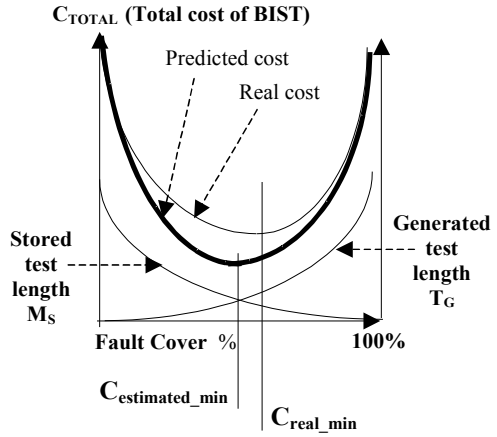


Fig. 7-7. Optimization of hybrid BIST

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. 7-7 shows a graphical solution for finding the optimum of mixing pseudorandom and stored test approaches as the trade-off between the memory cost and testing time. Let have the whole cost of the BIST to be found as

$$C_{TOTAL} = C_{TIME} + C_{HW} = \alpha TG + \beta MS$$

where C_{TIME} is the cost related to the time needed for test, C_{HW} is the hardware cost related to the BIST architecture, TG is the length of the test generated by LFSR, MS 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. 7-7, except the generated test length TG. The practical way would be in trying to find the curve for MS 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.

7.2 DefSim – a real life defect simulation environment

Many tools exist for behavioral emulation of defects (using mathematical models) but only few of them let students make real-life measurements. To fill this gap a project was started, the goal set on designing a chip which is, basically, full of defects. At first, a selection of different logic elements – simple and complex gates and circuits was chosen. In the second stage a set of different defects was selected, based on probabilistic analysis. Also, all stuck-at faults were included although the probability of such a fault tended to be quite small.
Each of the selected defects was implemented in standalone circuit (called cell), embedded in wrapper circuits. For each cell an power control unit, current monitor, input and output buffers were added. Design solution like this clears many problems like:

- Overcurrent protection system. Although the cells are 'defective from the start' we surely do not want to completely destroy them when activated.
- Multiple fault interaction elimination. Usage of switching logic for defect activation may cause side-effects on measurement results, thus

one fault per circuit should make it behave more naturally. This is especially important for shorts because every milli-ohm counts there.

- Overall circuit power reduction. The final product is connected via USB interface, thus the design is limited in power usage (about 100mA up to 500mA in special cases). Because of the wrapper logic it is possible to activate single cells at a time, considerably reducing the power consumption.
- Precise current detection for I_{ddq} testing and output voltage measurement.

Author's main contribution to this project was the command line interface utility, but he also participated in general chip design, modified the DefSim Linux driver for the use on 64-bit systems and helped with the Linux web server installation and administration.

7.2.1 DefSim user interface

DefSim is designed to be accessible through different means. The cheapest (per student) solution is server-based [116], utilizing Apache Tomcat web server and is accessible by every JavaScript enabled web browser [105]. Also, a local installation is available where DefSim measurement box is connected directly to each workplace (see diagram on Fig. 7-8) and can be accessed using graphical user interface or simply by using command line interface (CLI) DefSim utility.

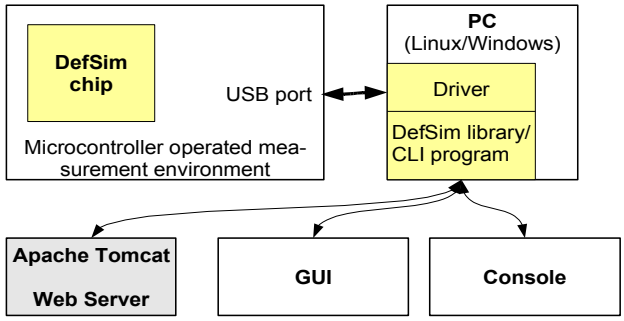


Fig. 7-8. DefSim system diagram

Web interface to DefSim measurement environment (called WebTT) is a multiuser solution where each user has to create (or log in to) its own account. Each user can create multiple test stimuli files and store them under different projects, alongside with the measurement result files for reports or further analysis.

Illustration 7-1 depicts login window to the DefSim measurement environment, also the DefSim measurement box and chip are visible.

Illustration 7-1. DefSim start page

7.2.2 Teaching CMOS defects on DefSim

A variety of different circuits with stuck-open or stuck-on transistor defects and classical SAF defects are implemented in the DefSim IC. Also two types of measurements can be used for detecting defects. Thus, various exercises are possible for students to work on (and not limited to) [13]:

- Simple simulation experiment. This experiment will show how logic functions may change in the presence of certain defect.
- Manual test generation and application. How to create tests for detecting certain defect and check if the test really detects it. This, of course expects a basic understanding of low level defect behavior and is a good way to learn defect modelling.
- Manual SAF test generation and application. A good way to show that the simple SAF model is not so good when dealing with real-life defects. This fact will become clear after applying SAF tests to all implemented defects.
- Diagnosis of SAF defects. When pre-generated tests are applied to certain circuit (with known function) and some of the circuit responses don't match with the expected truth table then it is possible to locate the defect using combinational diagnosis.
- Minimal test generation.

Test generation and compaction can be done either manually or by using ATPG tools from Turbo Tester package. All test stimuli insertion, analysis and use of ATPG toolset is incorporated into the web environment, so the users don't have to install them. A JavaScript-enabled web-browser is the only requirement for the user.

7.3 e-EDU student management system

An in-house design attempt to construct a prototype of powerful and versatile information system for different targets was started several years ago. As a result, three branches (views) emerged, respectively called as:

- ATI. (Illustration 7-2) This view displays many information pages about the department. Its well categorized links are easy to follow and everyone searching materials about Department of Computer Engineering (TUT) should find useful information.
- ITA. (Illustration 7-3) This is an administrative view, restricted to the public access and modifying of ATI and e-EDU views is possible through this view. Also, teachers side of e-EDU student management system is located there. Users with high level administrative rights can also affect ITA itself.
- EDU. (Illustration 7-4) This is usually referred to as e-EDU [104] and it is accessible for registered users only. Registering system is tied to the external student account management system. Meaning, when students apply for their computer classes account activation, they also gain access to the e-EDU system.

Although being one system logically, all these three parts are residing on different web-servers. This greatly simplifies user access checking and allows hiding of components the users are not supposed to see, even by mistake or by hacking attempt. Blocking access to one or the other view is also easy in this solution. Thanks to innovative modular architecture and flexible module management system this system is extremely adaptive and could be used for almost any task imaginable.

Illustration 7-2. ATI view

Illustration 7-3. ITA view with activated e-EDU administration page

7.3.1 e-EDU services

For students the e-EDU view enables access to many different items. Most important links are duplicated by icons in main subject view. Course material (both lectures and practicum guides), supervisor contacts and of course automatic practical work assignment system modules are associated to each subject.

Illustration 7-4. e-EDU, student view with activated subject page

The task assigment (in eEDU view for students), task managenent and student statistics (in ITA view) are incorporated into two modules developed by the author of this thesis.

The task assignment system is able to serve a range of worksheets – from simple static page up to complicated electronic worksheet. This means it can potentially be used for on-line examination using “interactive modules” in case the applets have additional built-in support for logging and reporting.

Completed work reports are then filed by e-EDU system and become visible to the teacher who then can accept, reject or send it back for improvement(s) using ITA view. Since all the reports are kept in a single storage it is easy to generate statistics about participating students and keep track of their course advancement and activity. The system can also handle typical situation where many students are divided into smaller groups with different supervisors and, possible, different tasks. Thus, the overall progress of whole course can be tracked in real-time.

7.4 Conclusions

The applet, described in Section 7.1 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.

Applets and other web services tied into one place like e-EDU make it very easy to access and use different resources, keeping close track of students progress.

Summary

In this thesis the following three results are presented:
- development of a new generic functional fault model to represent physical defects in digital circuits,
- development of methods for improving the quality of built-in self-testing in digital systems, and
- development of an educational applets based e-learning environment for learning and teaching the basics of testing digital systems.

A new generic functional fault model was proposed to increase the accuracy of evaluating the fault coverage of given test sets, and consequently improving the quality of testing including built-in self-testing. The fault model allows to map the physical transistor-level faults, like shorts or bridges, opens or any other logically describable defects on the higher gate-level or even on more higher macro logic level which reduces the complexity of fault simulation algorithms and helps to increase the speed of test quality analysis. It was shown also that the new fault model supports hierarchical approaches to test generation or fault analysis, since it can be regarded as a uniform interface for mapping faults from a given arbitrary level of abstraction to the next higher level of abstraction. The results on the new fault model are published in [15,18,19].

Different approaches were investigated to improve the efficiency and quality of built-in self-test architectures. First, hybrid BIST architectures, where pseudorandom test patterns were combined with deterministic test patterns, were researched. Then, a new functional hybrid BIST approach was developed where instead of pseudorandom test patterns the normal functional routines carried out in digital systems are combined with deterministic test patterns. And finally, a Design-for-Testability (DfT) technique was combined with Built-In Self-Test for sequential circuits to achieve the needed test quality.

For the hybrid BIST which combines pseudorandom test patterns with stored precomputed deterministic test patterns, a method and algorithms were developed for fast calculation of the cost of hybrid BIST. The cost can be calculated for different pseudorandom test sequences to find an optimal balance between pseudorandom and deterministic test sets, and to perform the hybrid self-test with minimum cost of both, time and memory, and without losing in test quality. Compared to the previous approaches, in this work a new, extremely fast procedure was proposed, which calculates costs on the basis of fault table

manipulations. Experiments on the ISCAS benchmark circuits showed that the new procedure is about two orders of magnitude faster than the compared one. The results on the hybrid BIST are published in [3,20,21,23-25].

The hybrid functional BIST (HyBIST) approach where the functional routines carried out in digital systems are combined with deterministic test patterns was investigated in the case of testing microprogrammed data-paths in digital systems. In the first test phase only the functional resources of a system are used for testing purposes. A functional microprogram is executed to control the data-path based on some very small deterministic input data. A response compressor like signature analyzer is connected to the data path to monitor the process. To guarantee a high test coverage for BIST, the second phase of the test is used which consists of applying additional deterministic test patterns pre-generated by an ATPG to test the random-pattern-resistant faults. A method was proposed to find the trade-off between the functional test and deterministic test parts. Experimental part of the work demonstrated the feasibility of the approach, and the advantage of combining functional and deterministic test patterns compared to the pure deterministic test. The results on the hybrid functional BIST are published in [12].

A Design-for-Testability (DfT) technique of Built-In Self-Test (BIST) for digital systems consisting of control and data parts was developed. The technique is based on making the status signals entering the control part controllable during the test mode to force the system under test to traverse all the branches in the FSM state transition graph. Extra outputs are added to the system under test in order to observe the values of the status bits masked out. This type of architecture requires little chip area overhead since a simple controller can be implemented to manipulate the control signals. The experimental research showed that simple LFSR does not provide an acceptable fault coverage for sequential designs. However, no universally best test generating approach was identified and the optimal solution appears to be highly dependent on designs pseudo-random testability characteristics. To improve the test quality if needed the hybrid BIST approach discussed above can be used. The results on improving the BIST by DFT are published in [13].

A set of tools (“interactive modules”) targeted to e-learning were developed for learning and teaching logic level test generation, built-in self-test, and fault diagnosis in digital circuits. The tools can support different university courses on computer engineering, switching and automata theories, digital electronics and design for testability to learn by hands-on exercises 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, how to locate a faulty gate, or how to design an optimal hybrid BIST

architecture 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, 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. The results on developing tools for e-Learning environment are published in [1-2, 4-11, 14, 16-19, 22].

Bibliography

[1] A. Al-Yamani, S. Mitra, E. J. McCluskey, "Optimized Reseeding by Seed Ordering and Encoding", IEEE Trans. CAD, pp. 264-270, 2005

[2] A. Jas, C. V. Krishna, N. A. Touba, "Weighted Pseudo-Random Hybrid BIST", IEEE Trans. VLSI Syst., pp. 1277-1283, 2004

[3] A. Markus, J. Raik, R. Ubar, "Test Set Minimization Using Bipartite Graphs", Proc. of the 6th Baltic Electronics Conference, Tallinn, Estonia, pp. 175-178, October 1998

[4] A. Thayse, "Boolean Calculus of Differences", Springer Verlag, 1981

[5] A. B. Kahng, S. Reda, "New and Improved BIST Diagnosis Methods From Combinatorial Group Testing Theory", IEEE Trans. on CAD of IC and Systems, pp. 533-542, March 2006

[6] A. Benso, S. Chiusano, S. Di Carlo, P. Prinetto, F. Ricciato, "HD2BIST: a Hierarchical Framework for BIST Scheduling, Data patterns delivering and diagnosis in SoCs", Proc. ITC, pp. 892-901, 2000

[7] A. P. Ströle, "BIST pattern generators using addition and subtraction operations", JETTA, pp. 69-80, Aug., 1977

[8] B. Könemann, "LFSR-Coded Test Patterns for Scan-Designs", Proc. European Test Conf, pp. 237-242, April 1991

[9] C. Fagot, O. Gascuel, P. Girard, C. Landrault, "On Calculating Efficient LFSR Seeds for Built-In Self-Test", Proc. of European Test Symposium, Munich, 1999

[10] C. Liu, K. Chakrabarty, M. Goessel, "An Interval-based Diagnosis Scheme for Identifying Failing Vectors in a Scan-BIST Environment", DATE, 2002

[11] C.-Y. Lee, C.-M. Li, "Segment Weighted Random BIST (SWR-BIST): A Low Power BIST Technique", Proc. ASSC, pp. 333-336, 2005

[12] D. Das, N. A. Touba, "Reducing test data volume using external/LBIST hybrid test patterns", Proc. Int. Test Conf., pp. 115-122, Oct. 2000

[13] D. Zhukov, "Development of an educational environment based on a new educational chip for “Diagnostics of Digital Systems” course.", TTU, 2004

[14] E. J. McCluskey, "Logic Design Principles: With Emphasis on Testable Semiconductor Circuits", Prentice Hall, 1986

[15] E. M. Rudnick, R. Vietti, A. Ellis, F. Corno, P. Prinetto, M. Sonza Reorda, "Fast sequential circuit test generation using high-level and gate-level techniques", Proc. of DATE, 1998

[16] E. B. Eichelberger, E. Lindbloom, "Random Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test", IBM J. Res. Dev., pp. 265-272, May 1983

[17] E. J. Marinissen, Y. Zorian, "Challenges in Testing Core-Based System ICs", IEEE Communications Magazine, pp. 104-109, June 1999

[18] F. Brglez, et al., "Hardware-Based Weighted Random Pattern Generation for Boundary-Scan", Proc. IEEE Int. Test Conf., pp. 264-274, 1989

[19] G. Jervan, A. Markus, P. Paomets, J. Raik, R. Ubar, "A CAD system for Teaching Digital Test", 2nd European Workshop on Microelectronics Education, Noordwijkerhout, the Netherlands, pp. 287-290, May 14-15, 1998

[20] G. Jervan, H. Kruus, E. Orasson, R. Ubar, "Hybrid BIST Optimization Using Reseeding and Test Set Compaction", Proc. of 10th EUROMICRO Conference on Digital System Design - DSD 2007, Lübeck, Germany, August 27 - 31, 2007

[21] G. Jervan, H. Kruus, E. Orasson, R. Ubar, "Optimization of Memory-Constrained Hybrid BIST for Testing Core-Based

Systems", IEEE 2nd International Symposium on Industrial Embedded Systems - SIES'2007, Lisbon, Portugal, 4-6 July 2007

[22] G. Jervan, P. Eles, Z. Peng, R. Ubar, M. Jenihhin, "Test Time Minimization for Hybrid BIST of Core-Based Systems", Proc. Asian Test Symp., pp. 318-323, Nov. 2003

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

[24] G. Kiefer, H.-J. Wunderlich, "Deterministic BIST with Multiple Scan Chains", Proc. Int. Test Conf, pp. 1057-1064, Oct. 1998

[25] G. Edirisooriya, J. P. Robinson, "Design of Low-Cost ROM Based Test Generators", Proc. VLSI Test Symp., pp. 61-66, April 1992

[26] G. Moore, "Cramming more components onto integrated circuits", Electronics, pp. 114-117, 1965

[27] H. Takahashi, Y. Tsugaoka, H. Ayano, Y. Takamatsu, "BIST Based Fault Diagnosis Using Ambiguous Test Set", Proc. 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, p. 80, 2003

[28] H. Tenhunen, "Invited talk: System-on-Chip Curriculum Challenges", Proc 5th European Workshop on Microelectronics Education – EWME, Lausanne Switzerland, April 15-16, 2003

[29] H.-C.Tsai, K.-T.Cheng, C.-J.Lin, S.Bhawmik, "Efficient Testpoint Selection for Scan-Based BIST", IEEE Trans. VLSI Syst., pp. 667–676, Dec. 1998

[30] H.-G. Liang, S. Hellebrand, H.-J. Wunderlich, "Two-Dimentional Test Data Compression for Scan-Based Deterministic BIST", Proc. Int. Test Conf., pp. 894-902, Sept. 2001

[31] H.-J. Wunderlich, "Multiple distributions for biased random test patterns", Proc. ITC, pp. 236-244, 1988

[32] H.-J. Wunderlich, "From Embedded Test to Embedded Diagnosis", IEEE 10th European Test Conference, Tallinn, , May 22-25, 2005

[33] H.-J. Wunderlich, G. Kiefer, "Bit-Flipping BIST", Proc. ICCAD, pp. 337-343, November 1996

[34] H. H. Chen, "Hierarchical BIST for SOC Design", Proc. Emerging Information Technology Conf., pp. 1-3, 2005

[35] H. Kruus, E. Orasson, T. Robal, R. Ubar, "Investigating Defects in Digital Circuits by Boolean Differential Equations", The 4th International Conference “Distance Learning – Educational Sphere of XXI Century” (DLESC’04), Minsk, pp. 432-435, November 10-13, 2004

[36] I. Pomeranz, S. M. Reddy, "Built-in test generation for synchronous sequential circuits", ICCAD, pp. 421-426, 1997

[37] I. Voyiatzis, C. Halatsis, "A Low-Cost Concurrent BIST Scheme for Increased Dependability", IEEE Trans. on Dependable and Secure Computing, pp. 150-156, April-June 2005

[38] J. Ghosh-Dastidar, N. A. Touba, "A Rapid and Scalable Diagnosis Scheme for BIST Environments with a Large Number of Scan Chains", VTS, 2000

[39] J. Rajski, J. Tyszer, "Multiplicative window generators of pseudorandom test vectors", Proc. European Design and Test Conf., pp. 42-48, 1996

[40] J. Rajski, J. Tyszer, "Arithmetic BIST For Embedded Systems", Prentice-Hall, NJ, 1998

[41] J. Rajski, J. Tyszer, N. Zacharia, "Test Data Decompression for Multiple Scan Designs with Boundary Scan", IEEE Trans. Comput., pp. 1188-1200, 1998

[42] J. Hartmann, G. Kemnitz, "How to Do Weighted Random Testing for BIST", Proc. IEEE Int. Conf. Comput.-Aided Design, pp. 568–571, 1993

[43] J. M. Soden, C. F. Hawkins, "Quality Testing Requires Quality Thinking", Proc. Int. Test Conference, pp. 596, 1993

[44] J. P. Shen, W. Maly, J. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits", IEEE Design and Test, pp. 13-26, 1985

[45] J. Waicukauski, E. Lindbloom, E. Eichelberger, O. Forlenza, "A Method for Generating Weighted Random Test Patterns", IEEE Trans. Comput., pp. 149–161, Mar. 1989

[46] K. Chakrabarty, B. T. Murray, V. Iyengar, "Deterministic built-in test pattern generation for high-performance circuits using twisted-ring counters", IEEE Trans. VLSI Systems, Oct. 2000

[47] K. Sekar, S. Dey, "LI-BIST: A Low-Cost Self-Test Scheme for SoC Logic Cores and Interconnects", Proc. 20th IEEE VTS, pp. 1-6, April 2002

[48] L. Lei, K. Chakrabarty, "Hybrid BIST on Repeating Sequences and Cluster Analysis", Proc. DATE, pp. 1142-1147, March 2005

[49] L. Nachman, K. K. Saluja, S. Upadhyaya, R. Reuse, "Random pattern testing for sequential circuits revisited", Proc. of FTCS-26, 1996

[50] L.-T. Wang, Ch.-W. Wu, X. Wen, "VLSI Test Principles and Architectures. Design for Testability", Morgan Kaufmann Publishers, 2006

[51] M. Abramovici, M. Breuer, A.D. Friedman, "Digital System Testing and Testable Design", Computer Sci. Press, 1995

[52] M. Abramovici, Ch. E. Stroud, J. M. Emmert, "Online BIST and BIST-Based Diagnosis of FPGA Logic Blocks", IEEE Trans. on VLSI Systems, pp.1284-1294, Dec. 2004

[53] M. Abramovici, Ch. E. Stroud, J. M. Emmert, "Online BIST and BIST-Based Diagnosis of FPGA Logic Blocks", Proc. 18th IEEE DFT, pp. 1-8, 2003

[54] M. Chatterjee, D. K. Pradhan, "A novel pattern generator for near-perfect fault-coverage", VLSI Test Symposium, pp. 417-425, 1995

[55] M. Chatterjee, D. K. Pradhan, W. Kunz, "LOT: Logic optimization with testability – New transformations using recursive learning", Proc. Int. Conf. on CAD, pp. 318-325, Nov. 1995

[56] M. Roper, "Software Testing", McGraw-Hill Book Company, 1994

[57] M. Sugihara, H. Date, H. Yasuura, "A Novel Test Methodology for Core-Based System LSIs and a Testing Time Minimization Problem", Proc. Int. test Conf., pp. 465-472, Oct 1998

[58] 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", Proc. EWDTC'03, Scientific-Technical Journal Radioelectronics and Informatics, pp. 69-73, July-Sept. 2003

[59] M. Abramovici, M. Breuer, A. D. Friedman, "Digital System Testing and Testable Design", Computer Sci. Press, 1995

[60] M. E. Aboulhamid, E. Cerny, "A class of test generators for Built-In Testing", IEEE Trans. Comput., pp. 957-959, 1983

[61] M. F. AlShaibi, Ch. Kime, "MFBIST: A BIST Method for Random Pattern Resistant Circuits", Proc. ITC, pp. 176-185, Oct. 1996

[62] M. Jacomet, W. Guggenbuhl, "Layout-Dependent Fault Analysis and Test Synthesis for CMOS Circuits", IEEE Trans. on CAD, pp. 888-899, 1993

[63] M. L. Bushnell, "Increasing Test Coverage in a VLSI Design Course", ITC, Atlantic City, NJ, USA, p. 1133, 1999

[64] N. Tamarapalli, J. Rajski, "Constructive multi-phase test point insertion for scan-based BIST", Proc. ITC, pp. 649-658, Oct. 1996

[65] N. Zacharia, J. Rajski, J. Tyszer, "Decompression of Test Data Using Variable-Length Seed LFSRs", Proc. of VLSI Test Symposium, pp. 426-433, 1995

[66] N. A. Touba, E. J. McCluskey, "Bit-Fixing in Pseudorandom Sequences for Scan BIST", IEEE Trans. CAD, pp. 545-555, 2001

[67] N. A. Touba, E. J. McCluskey, "Transformed Paseudo-Random Patterns for BIST", Proc. VLSI Test Symp., pp. 410-416, April 1995

[68] N. A. Touba, E. J. McCluskey, "Test Point Insertion Based on Path Tracing", Proc. VLSI Test Symp., pp. 2-8, 1996

[69] N. Z. Basturkmen, S. M. Reddy, I. Pomeranz, "Pseudo Random Patterns Using Markov Sources for Scan BIST", Proc. IEEE Int. Test Conf., pp.1013–1021, 2002

[70] P. Prinetto et al., "Panel Session: Design & Test Education and Training in Europe", 7th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems – DDECS, Stara Lesna, Slovakia, , April 18-21, 2004

[71] P. D. Hortensius, et al., "Cellular automata based pseudorandom number generators for BIST", IEEE Trans. CAD, pp. 842-859, 1990

[72] P. H. Bardell, W. H. McAnney, J.Savir, "Built-In Test for VLSI: Pseudo-Random Techniques", J. Wiley & Sons, 1987

[73] P. H. Bardell, W. H. McAnney, "Self-Testing of multiple logic modules", Proc. Int.Test Conference, pp. 200-204, Oct. 1982

[74] P.Nigh and W.Maly, "Layout-Driven Test Generation", Proc.1989 International Conference on Computer Aided Design, pp. 154-157, 1989

[75] R. Dorsch, H-J. Wunderlich, "Accumulator Based Deterministic BIST", ITC, Washington D.C, 1998

[76] R. Dorsch, H.-J. Wunderlich, "Tailoring ATPG for Embedded Testing", Proc. Int. Test Conf., pp. 530-537, Oct. 2001

[77] R. Ubar, A. Jutman, E. Orasson, J. Raik, T. Evartson, H.-D. Wuttke, "Internet-Based Software for Teaching Test of Digital Circuits", "Microelectronics Education", Marcombo Boixareu Ed., 2002

[78] R. Ubar, A. Jutman, M. Kruus, E. Orasson, S. Devadze, H.-D. Wuttke, "Learning Digital Test and Diagnostics via Internet", International Journal of Emerging Technologies in Learning. International Journal of Online Engineering, pp. 1-9, 2007

[79] 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, CIT-6, pp. 1-6, March 26-28, 2003

[80] R. Ubar, E. Orasson, H.-D. Wuttke, "Interactive Teaching Software "Introduction To Digital Test"", 45th International Conference, Ilmenau (Germany), pp. 949-954, October 4-6, 2000

[81] R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke, "Combining Learning, Training and Research in Laboratory Course for Design and Test", 7th Baltic Electronics Conference, Tallinn, pp. 221-224, October 8-11, 2000

[82] R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke, "Teaching Advanced Test Issues in Digital Electronics. Summary", 6th IEEE International Conference on Information Technology Based Higher Education and Training, Santo Domingo, pp. 46-47, July 7-9, 2005

[83] R. Ubar, G. Jervan, H. Kruus, E. Orasson, I. Aleksejev, "Optimization of the Store-and-Generate Based Built-in Self-Test", Baltic Electronics Conference, Laulasmaa, Estonia, pp. 199-202, Oct. 2006

[84] R. Ubar, G. Jervan, Z. Peng, E. Orasson, R. Raidma, "Fast Test Cost Calculation for Hybrid BIST in Digital Systems", Proc. of EUROMICRO Symposium on Digital Systems Design, Warsaw, pp. 318-325, September 4-6, 2001

[85] R. Ubar, H.-D. Wuttke, "Action-Based Learning System for Teaching Digital Electronics and Test", 3rd European Workshop on Microelectronics Education, Aix en Provence, France, May 18-19, 2000

[86] R. Ubar, N. Mazurova, J. Smahtina, E. Orasson, J. Raik, "HyFBIST: Hybrid Functional Built-In Self-Test in Microprogrammed Data-Paths of Digital Systems", Int. Conference MIXDES, Szczecin, pp. 497-502, June 24-26, 2004

[87] R. Ubar, S. Kostin, J. Raik, "Experimental Comparison of Different Diagnosis Algorithms in the BIST Environment", 8th IEEE Latin-American Test Workshop, March, 2007

[88] R. Ubar, T. Shchenova, G. Jervan, Z. Peng, "Energy Minimization for Hybrid BIST in a System-on-Chip Test Environment", Proc. of 10th IEEE European Test Symposium, Tallinn, pp. 2-7, May 22-25, 2005

[89] R. Ubar. E. Orasson, H.-D. Wuttke, "Internet-Based Software for Teaching Test of Digital Circuits", 23rd Int. Conf. on Microelectronics, Nis, Yugoslavia, Vol. 2, pp. 659-662, May 12-15, 2002

[90] R. Dandapani, J. Patel, J. Abraham, "Design of Test Pattern Generators for Built-In Test", Proc. Int. Test Conf., pp. 315-319, Oct. 1984

[91] R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke, "Teaching Advanced Test Issues in Digital Electronics", 6th IEEE International Conference on Information Technology Based Higher Education and Training, Santo Domingo, pp. S2B-5 – S2B-10, July 7-9, 2005

[92] S. Chiusano, S. Di Carlo, P. Prinetto, H.-J. Wunderlich, "On applying the set covering model to reseeding", Proc. DATE, pp. 156-161, 2001

[93] S. Devadze, A. Jutman, A. Sudnitson, R. Ubar, H.-D. Wuttke, "Teaching Digital RT-Level Self-Test Using a Java Applet", NORCHIP'2002 Conf., Copenhagen, pp. 322-328, November 11-12, 2002

[94] S. Hellebrand, B. Reeb, S. Tarnick, H.-J.Wunderlich, "Pattern Generation for a Deterministic BIST Scheme", Proc. Int. Conf. on CAD, pp. 88-94, Nov. 1995

[95] S. Hellebrand, H.-J. Wunderlich, A. Hertwig, "Mixed-Mode BIST Using Embedded Processors", No. 12, Journal of Electronic Testing: Theory and Applications, pp. 127-138, 1998

[96] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataramann, B. Courtois, "Built-in Test for Circuits with Scan Based on Reseeding of Multi-Polynomial LFSR", IEEE Trans. on Comput, pp. 223-233, Feb. 1995

[97] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of Vector Patterns through Reseeding of Multiple-Polynomial Linear

Feedback Shift Registers", Proc. IEEE Int. Test Conf., pp. 120-129, 1992

[98] S. R. Rao, B. Y. Pan, J. R. Armstrong, "Hierarchical Test Generation for VHDL Behavioral Models", Proc. EDAC, pp. 175-183, Feb. 1993

[99] S. Swaminathan, K. Chakrabarty, "A Deterministic Scan-BIST Architecture with Application to Field Testing of High-Availability Systems", IEEE Custom Integrated Circuits Conf., pp. 259-262, 2001

[100] S. Wang, "A BIST TPG for Low Power Dissipation and High Fault Coverage", Proc. Int. Test Conf., Baltimore, MD, pp. 777-789, Oct. 30 - Nov. 1, 2001

[101] S. Zhang, M. Choi, F. Lombardi, "Cost-Driven Optimization of Coverage of Combined Built-In Self-Test/Automated Test Equipment Testing", IEEE Trans. on Instrumentation and Measurement, pp. 1094-1100, June 2007

[102] S. Jain, J. Abraham, S.-K. Vooka, S. Kale, A. Dutta, R. Parekhji, "Enhancements in Deterministic BIST Implementations for Improving Test of Complex SOCs", Proc. 20th Int. Conf. on VLSI Design, pp. 1-6, 2007

[103] "The International Technology Roadmap for Semiconductors. 2005 Edition (ITRS 2005)", http://public.itrs.net/

[104] T. Robal, A. Kalja, "e-EDU. An Information System for e-learning Services", Vol 118, Databases and Information Systems, J Barzdins and A. Caplinskas (Eds). Frontiers in Artificial Intelligence and Applications, Amsterdam, The Netherlands, IOS Press, pp. 288-298, 2005

[105] V. Vislogubov, A. Jutman, H. Kruus, E. Orasson, J. Raik, R. Ubar, "Diagnostic Software with WEB Interface for Teaching Purposes", Proc. of the 9th Biennial Baltic Electronics Conference, Tallinn, pp. 255-258, Oct. 3-6, 2004

[106] V. D. Agrawal, "Increasing Test Coverage in a VLSI Design Course", International Test Conference, Atlantic City, NJ, USA, p. 1131, 1999

[107] V. K. Agrawal, E. Cerny, "Store and Generate Built-In Testing Approach", Digest of Papers, Fault-Tolerant Computing Symp, pp. 35-40, June 1981

[108] W. Li, C. Yu, S. M. Reddy, I. Pomeranz, "A Scan BIST Generation Method Using a Markov Source and Partial BIST Bit-Fixing", Proc. IEEE-ACM Design Autom. Conf., pp. 554–559, 2003

[109] W. Maly, J. P .Shen, J. Ferguson, "Systematic Characterization of Physical Defects for Fault Analysis of MOS IC Cells", Proc. 1984 International Test Conference, Philadelphia, pp. 390-399, 1984

[110] Y. Nakamura, J. Savir, H. Fujiwara, "BIST Pretest of ICs: Risks and Benefits", Proc. 24th IEEE VTS, May 2006

[111] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices", Proc. VLSI Testing Symp, pp. 4–9, 1993

[112] Y. Zorian, E.J. Marinissen, S. Dey, "Testing embedded core-based system chips", Proc. IEEE International Test Conference. Washington, pp. 130–143, 1998

[113] Y.-C. Lin, F. Lu, K.-T. Cheng, "Pseudo-Functional Scan-based BIST for Delay Fault", Proc. 23nd IEEE VTS, pp. 1-6, May 2005

[114] Z. Zhao, B. Pouya, N. A. Touba, "BETSY: Synthesizing Circuits for a Specified BIST", Proc. ITC, pp. 144-153, Oct. 1998

[115] Zorian, Y., Marinissen, E. J. and Dey, S., "Testing embedded core-based system chips", Proc. IEEE International Test Conference. Washington, pp. 130–143, 1998

[116] "DefSim Home Page", http://www.defsim.com

[117] "VILAB", http://vilab.dcs.elf.stuba.sk/

[118] "High Level Benchmarks", http://www.cbl.ncsu.edu:16080/benchmarks/HLSynth92/

[119] "Turbo Tester Reference Manual", http://www.pld.ttu.ee/tt

Appendix

Curriculum Vitae

1. Personal Data

First name	Elmet
Last name	Orasson
Date and place of birth	18.06.1974, Võru
Marital status	single
Children	no

2. Contacts

Address	Luha 23-11, Võru, 65607 Estonia
	Tsolgo, Võrumaa, 65552 Estonia
Phone	+372-620-2263
	+372-56683-512
	+372-78-60023
E-mail	elmet@pld.ttu.ee

3. Education

Educational Institution	Aeg	Haridus
Tallinn University of Technology	2002-2007	Information and Communication Technology/PhD studies
Tallinn University of Technology	2001-2002	Computer and Systems Engineering/MsC
Tallinn University of Technology	1992-2001	Computer and Systems Engineering/Dip.Eng
Võru Kreutzwald Secondary Scool (Võru Kreutzwald Gymnasium)	1989 - 1992	Secondary Education

4. Language Skills

Estonian (formal)	high level
Estonian/võro	native
English	high level
Finnish	intermediate
German	intermediate
Russian	poor

5. Special Courses

6. Career

2005 - ...	Tallinn University of Technology, Faculty of Infotechnology, Department of Computer Engineering, Chair of Computer Engineering and Diagnostics	Researcher
2001 - 2005	Tallinn University of Technology, Faculty of Infotechnology, Department of Computer Engineering, Chair of Computer Engineering and Diagnostics	Senior engineer
1998 - 2001	Tallinn University of Technology, Faculty of Infotechnology, Department of Computer Engineering, Chair of Computer Engineering and Diagnostics	Engineer
1994 - 1998	Tallinn University of Technology, Faculty of Infotechnology, Department of Computer Control, Chair of Circuit Theory and Design	Engineer

7. Scientific Work

Publications

1. R. Ubar, E.Orasson, H.-D. Wuttke. “Interactive Teaching Software 'Introduction To Digital Test' “. 45th International Conference, Ilmenau (Germany), October 4-6, 2000, pp. 949-954.

2. R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke. “Combining Learning, Training and Research in Laboratory Course for Design and Test”. 7th Baltic Electronics Conference, Tallinn, October 8-11, 2000, 221-224.

3. R. Ubar, G. Jervan, Z. Peng, E. Orasson, R. Raidma. “Fast Test Cost Calculation for Hybrid BIST in Digital Systems”. Proc. of EUROMICRO Symposium on Digital Systems Design, Warsaw, September 4-6, 2001, pp. 318-325.

4. R. Ubar, E. Orasson, T. Evartson. “Java Applet for Self-Learning of Digital Test Issues”. 13th EAEEIE Conference, York, Great Britain, April 8-10, 2002

5. R. Ubar. E. Orasson, H.-D. Wuttke. “Internet-Based Software for Teaching Test of Digital Circuits”. 23rd Int. Conf. on Microelectronics. Nis, Yugoslavia, May 12-15 2002, Vol.2, pp. 659-662.

6. 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.

7. R. Ubar, E. Orasson, T. Evartson. “Self-learning tool for digital test”. Proceedings of 2nd Int. Conf. “Distance learning – educational sphere of the XXI century”, Minsk, Belarus, Nov. 26-28, 2002, pp. 36-38.

8. 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.

9. 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. Summaries. Sousse, Tunisia, March 26-28, 2003, CIT-6, pp. 134.

10. 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”. Journal of Radioelectronics and Informatics, No3 (24), July – September, 2003, pp. 69-73.

11. S. Devadze, R. Gorjachev, A. Jutman, E. Orasson, V. Rosin, R. Ubar. “E-Learning Tools for Digital Test”. In “Distance Learning – Educational Environment of the XXI Century”, Minsk, 2003, pp. 336-342.

12. R. Ubar, N. Mazurova, J. Smahtina, E. Orasson, J. Raik. “HyFBIST: Hybrid Functional Built-In Self-Test in Microprogrammed Data-Paths of Digital Systems”. Int. Conference MIXDES, Szczecin, June 24-26, 2004, pp. 497-502.

13. J. Raik, E. Orasson, R. Ubar. “Sequential Circuits BIST with Status BIT Control”. Int. Conference MIXDES, Szczecin, June 24-26, 2004, pp. 507-510.

14. E. Ivask, A. Jutman, E. Orasson, J. Raik, R. Ubar, H-D. Wuttke. “Research Environment for Teaching Digital Test”. 49. Int. Conf. IWK, Ilmenau, Germany, September 27-30, 2004, pp. 468-473.

15. H. Kruus, E. Orasson, T. Robal, R. Ubar. “Investigating Defects in Digital Circuits by Boolean Differential Equations”. The 4th International Conference “Distance Learning – Educational Sphere of XXI Century” (DLESC’04), Minsk, November 10-13, 2004, pp. 432-435.

16. V. Vislogubov, A. Jutman, H. Kruus, E. Orasson, J. Raik, R. Ubar. “Diagnostic Software with WEB Interface for Teaching Purposes”. Proc. of the 9th Biennial Baltic Electronics Conference, Oct. 3-6, 2004, Tallinn, pp. 255-258

17. A. Jutman, J. Raik, E. Orasson, R. Ubar. Overview of the Educational Tools developed in REASON. Workshop on Research and Training Action for System on Chip DesigN – REASON, Tallinn, May 21, 2005, 7 p.

18. R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke. “Teaching Advanced Test Issues in Digital Electronics”. 6th IEEE International Conference on Information Technology Based Higher Education and Training. July 7-9, 2005, Santo Domingo, pp. S2B-5 – S2B-10.

19. R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke. “Teaching Advanced Test Issues in Digital Electronics. Summary”. 6th IEEE International Conference on Information Technology Based Higher Education and Training. July 7-9, 2005, Santo Domingo, pp. 46-47.

20. E. Orasson. “E-learning software for digital test”. IKDK annual conference pp. 133 – 134. TUT, 2006

21. R. Ubar, G. Jervan, H. Kruus, E. Orasson, I. Aleksejev. Optimization of the Store-and-Generate Based Built-in Self-Test. IKTDK annual conference. ISBN 9985-59-624-2. Jäneda, Estonia, 12.-13. may 2006, pp. 93-96.

22. R. Ubar, G. Jervan, H. Kruus, E. Orasson, I. Aleksejev. “Optimization of the Store-and-Generate Based Built-in Self-Test”. Baltic Electronics Conference. Laulasmaa, Oct. 2006, pp. 199-202.

23. R. Ubar, A. Jutman, M. Kruus, E. Orasson, S. Devadze, H.-D. Wuttke. “Learning Digital Test and Diagnostics via Internet”. International Journal of Emerging Technologies in Learning. International Journal of Online Engineering, Vol. 3, No. 1, pp.1-9, 2007.

24. H. Kruus, G. Jervan, E. Orasson, R. Ubar. “Optimization of Memory-Constrained Hybrid BIST for Testing Core-Based Systems”. IKTDK 2007 annual conference. Viinistu, Estonia, May 11-12, 2007, pp. 133-136.

25. G. Jervan, H. Kruus, E. Orasson, R. Ubar. “Optimization of Memory-Constrained Hybrid BIST for Testing Core-Based Systems”. IEEE 2nd International Symposium on Industrial Embedded Systems - SIES'2007. Lisbon, Portugal, 4-6 July 2007.

26. G. Jervan, H. Kruus, E. Orasson, R. Ubar. Hybrid BIST Optimization Using Reseeding and Test Set Compaction. Proc. of 10th EUROMICRO Conference on Digital System Design - DSD 2007, Lübeck, Germany, August 27 - 31, 2007.

27. R. Ubar, A. Jutman, M. Kruus, E. Orasson, S. Devadze, H.-D. Wuttke, “Learning Digital Test and Diagnostics via Internet”. International Journal of Online Engineering, 3(1), 1 – 9. 2007

28. G. Jervan, H. Kruus, E. Orasson, R. Ubar, “Optimization of Memory-Constrained Hybrid BIST for Testing Core-Based Systems”. Proceedings of the IEEE 2nd International Symposium on Industrial Embedded Systems - SIES'2007, Lisbon, Portugal, 4-6 July 2007. IEEE Computer Soc, 2007, 71 – 77.

8. Accomplished and defended theses

Elmet Orasson, Master's Degree, 2002, (sup) Raimund Ubar, “Development of algorithms for test analysis”, Tallinn University of Technology, Faculty of Infotechnology, Department of Computer Engineering

9. Research Interests

Natural Sciences and Engineering, Electrical Engineering and Electronics

10. Other research projects

Elulookirjeldus

1. Isikuandmed

Eesnimi	Elmet
Perekonnanimi	Orasson
Kodakondsus	Eesti
Sünniaeg (pp.kk.aaaa) ja koht	18.06.1974, Võru
Perekonnaseis	vallaline
Lapsed	ei ole

2. Kontaktandmed

Aadress	Luha 23-11, Võru, 65607 Eesti Tsolgo, Võrumaa, 65552 Eesti
Telefon	+372-620-2263 +372-56683-512 +372-78-60023
E-post	elmet@pld.ttu.ee

3. Haridustee

Õppeasutus	Aeg	Haridus
Tallinna Tehnikaülikool	2002-2007	Doktorantuur, TTÜ
Tallinna Tehnikaülikool	2001-2002	Tehnikateaduste magister, TTÜ
Tallinna Tehnikaülikool	1992-2001	Diplomiinsener, Arvuti- ja süsteemitehnika õppevaldkond, TTÜ
Fr. R. Kreutzwaldi nim. Võru I Keskkool (Võru Kreutzwaldi Gümnaasium)	1989 - 1992	keskharidus

4. Keelteoskus

Eesti (kirjakeel)	kõrgtase
Eesti (võro keel)	emakeel
Inglise	kõrgtase
Soome	kesk
Saksa	kesk
Vene	alg/kesk

5. Täiendõpe

6. Teenistuskäik

2005 - ...	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutitehnika instituut, Arvutitehnika- ja diagnostika õppetool	Teadur
2001 - 2005	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutitehnika instituut, Arvutitehnika- ja diagnostika õppetool	Vaneminsener
1998 - 2001	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutitehnika instituut, Arvutitehnika- ja diagnostika õppetool	Insener
1994 - 1998	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Automaatikainstituut, Siduteooria ja -disaini õppetool	Insener

7. Teadustegevus

Publikatsioonid

1. R. Ubar, E.Orasson, H.-D. Wuttke. “Interactive Teaching Software 'Introduction To Digital Test' “. 45th International Conference, Ilmenau (Germany), October 4-6, 2000, pp. 949-954.

2. R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke. “Combining Learning, Training and Research in Laboratory Course for Design and Test”. 7th Baltic Electronics Conference, Tallinn, October 8-11, 2000, 221-224.

3. R. Ubar, G. Jervan, Z. Peng, E. Orasson, R. Raidma. “Fast Test Cost Calculation for Hybrid BIST in Digital Systems”. Proc. of EUROMICRO Symposium on Digital Systems Design, Warsaw, September 4-6, 2001, pp. 318-325.

4. R. Ubar, E. Orasson, T. Evartson. “Java Applet for Self-Learning of Digital Test Issues”. 13th EAEEIE Conference, York, Great Britain, April 8-10, 2002

5. R. Ubar. E. Orasson, H.-D. Wuttke. “Internet-Based Software for Teaching Test of Digital Circuits”. 23rd Int. Conf. on Microelectronics. Nis, Yugoslavia, May 12-15 2002, Vol.2, pp. 659-662.

6. 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.

7. R. Ubar, E. Orasson, T. Evartson. “Self-learning tool for digital test”. Proceedings of 2nd Int. Conf. “Distance learning – educational sphere of the XXI century”, Minsk, Belarus, Nov. 26-28, 2002, pp. 36-38.

8. 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.

9. 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. Summaries. Sousse, Tunisia, March 26-28, 2003, CIT-6, pp. 134.

10. 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”. Journal of Radioelectronics and Informatics, No3 (24), July – September, 2003, pp. 69-73.

11. S. Devadze, R. Gorjachev, A. Jutman, E. Orasson, V. Rosin, R. Ubar. “E-Learning Tools for Digital Test”. In “Distance Learning – Educational Environment of the XXI Century”, Minsk, 2003, pp. 336-342.

12. R. Ubar, N. Mazurova, J. Smahtina, E. Orasson, J. Raik. “HyFBIST: Hybrid Functional Built-In Self-Test in Microprogrammed Data-Paths of Digital Systems”. Int. Conference MIXDES, Szczecin, June 24-26, 2004, pp. 497-502.

13. J. Raik, E. Orasson, R. Ubar. “Sequential Circuits BIST with Status BIT Control”. Int. Conference MIXDES, Szczecin, June 24-26, 2004, pp. 507-510.

14. E. Ivask, A. Jutman, E. Orasson, J. Raik, R. Ubar, H-D. Wuttke. “Research Environment for Teaching Digital Test”. 49. Int. Conf. IWK, Ilmenau, Germany, September 27-30, 2004, pp. 468-473.

15. H. Kruus, E. Orasson, T. Robal, R. Ubar. “Investigating Defects in Digital Circuits by Boolean Differential Equations”. The 4th International Conference “Distance Learning – Educational Sphere of XXI Century” (DLESC’04), Minsk, November 10-13, 2004, pp. 432-435.

16. V. Vislogubov, A. Jutman, H. Kruus, E. Orasson, J. Raik, R. Ubar. “Diagnostic Software with WEB Interface for Teaching Purposes”. Proc. of the 9th Biennial Baltic Electronics Conference, Oct. 3-6, 2004, Tallinn, pp. 255-258

17. A. Jutman, J. Raik, E. Orasson, R. Ubar. Overview of the Educational Tools developed in REASON. Workshop on Research and Training Action for System on Chip DesigN – REASON, Tallinn, May 21, 2005, 7 p.

18. R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke. “Teaching Advanced Test Issues in Digital Electronics”. 6th IEEE International Conference on Information Technology Based Higher Education and Training. July 7-9, 2005, Santo Domingo, pp. S2B-5 – S2B-10.

19. R. Ubar, E. Orasson, J. Raik, H.-D. Wuttke. “Teaching Advanced Test Issues in Digital Electronics. Summary”. 6th IEEE International Conference on Information Technology Based Higher Education and Training. July 7-9, 2005, Santo Domingo, pp.46-47.

20. E. Orasson. “E-learning software for digital test”. IKDK aastakonverentsi artiklite kogumik (133 – 134). Tallinna Tehnikaülikool, 2006

21. R. Ubar, G. Jervan, H. Kruus, E. Orasson, I. Aleksejev. Optimization of the Store-and-Generate Based Built-in Self-Test. Info- ja kommunikatsioonitehnoloogia doktorikooli IKTDK aastakonverentsi kogumik. ISBN 9985-59-624-2. Jäneda, 12.-13. mai 2006, lk. 93-96.

22. R. Ubar, G. Jervan, H. Kruus, E. Orasson, I. Aleksejev. “Optimization of the Store-and-Generate Based Built-in Self-Test”. Baltic Electronics Conference. Laulasmaa, Oct. 2006, pp.199-202.

23. R. Ubar, A. Jutman, M. Kruus, E. Orasson, S. Devadze, H.-D. Wuttke. “Learning Digital Test and Diagnostics via Internet”. International Journal of Emerging Technologies in Learning. International Journal of Online Engineering, Vol. 3, No. 1, pp.1-9, 2007.

24. H. Kruus, G. Jervan, E. Orasson, R. Ubar. “Optimization of Memory-Constrained Hybrid BIST for Testing Core-Based Systems”. IKTDK 2007 aastakonverents. Viinistu, Mai 11-12, 2007, pp. 133-136.

25. G. Jervan, H. Kruus, E. Orasson, R. Ubar. “Optimization of Memory-Constrained Hybrid BIST for Testing Core-Based Systems”. IEEE 2nd International Symposium on Industrial Embedded Systems - SIES'2007. Lisbon, Portugal, 4-6 July 2007.

26. G. Jervan, H. Kruus, E. Orasson, R. Ubar. Hybrid BIST Optimization Using Reseeding and Test Set Compaction. Proc. of 10th EUROMICRO Conference on Digital System Design - DSD 2007, Lübeck, Germany, August 27 - 31, 2007.

27. R. Ubar, A. Jutman, M. Kruus, E. Orasson, S. Devadze, H.-D. Wuttke, “Learning Digital Test and Diagnostics via Internet”. International Journal of Online Engineering, 3(1), 1 – 9. 2007

28. G. Jervan, H. Kruus, E. Orasson, R. Ubar, “Optimization of Memory-Constrained Hybrid BIST for Testing Core-Based Systems”. Proceedings of the IEEE 2nd International Symposium on Industrial Embedded Systems - SIES'2007, Lisbon, Portugal, 4-6 July 2007. IEEE Computer Soc, 2007, 71 – 77.

8. Kaitstud lõputööd

Diplomitöö	Java-põhine interaktiivne õppeprogramm “Sissejuhatus testi ja diagnostikasse”	2001
Magistritöö	Digitaalsüsteemide testi- ja diagnostikaalase õppetarkvara arendus	2002

9. Teadustöö põhisuunad

Loodusteadused ja tehnika, Elektrotehnika ja elektroonika (Digitaalsüsteemide istestimine)

10. Teised uurimisprojektid

Allkiri Kuupäev

DISSERTATIONS DEFENDED AT TALLINN UNIVERSITY OF TECHNOLOGY ON INFORMATICS AND SYSTEM ENGINEERING

1. **Lea Elmik**. Informational modelling of a communication office. 1992.
2. **Kalle Tammemäe**. Control intensive digital system synthesis. 1997.
3. **Eerik Lossmann**. Complex signal classification algorithms, based on the third-order statistical models. 1999.
4. **Kaido Kikkas**. Using the Internet in rehabilitation of people with mobility impairments - case studies and views from Estonia. 1999.
5. **Nazmun Nahar**. Global electronic commerce process: business-to-business. 1999.
6. **Jevgeni Riipulk**. Microwave radiometry for medical applications. 2000.
7. **Alar Kuusik**. Compact smart home systems: design and verification of cost effective hardware solutions. 2001.
8. **Jaan Raik**. Hierarchical test generation for digital circuits represented by decision diagrams. 2001.
9. **Andri Riid**. Transparent fuzzy systems: model and control. 2002.
10. **Marina Brik**. Investigation and development of test generation methods for control part of digital systems. 2002.
11. **Raul Land**. Synchronous approximation and processing of sampled data signals. 2002.
12. **Ants Ronk**. An extended block-adaptive Fourier analyser for analysis and reproduction of periodic components of band-limited discrete-time signals. 2002.
13. **Toivo Paavle**. System level modeling of the phase locked loops: behavioral analysis and parameterization. 2003.
14. **Irina Astrova**. On integration of obj ect-oriented applications with relational databases. 2003.
15. **Kuldar Taveter**. A multi-perspective methodology for agent-oriented business modelling and simulation. 2004.
16. **Taivo Kangilaski**. Eesti Energia käiduhaldussüsteem. 2004.
17. **Artur Jutman**. Selected issues of modeling, verification and testing of digital systems. 2004.

18. **Ander Tenno**. Simulation and estimation of electro-chemical processes in maintenance-free batteries with fixed electrolyte. 2004.
19. **Oleg Korolkov**. Formation of diffusion welded Al contacts to semiconductor silicon. 2004.
20. **Risto Vaarandi**. Tools and techniques for event log analysis. 2005.
21. **Marko Koort**. Transmitter power control in wireless communication systems. 2005.
22. **Raul Savimaa**. Modelling emergent behaviour of organizations. Time-aware, UML and agent based approach. 2005.
23. **Raido Kurel**. Investigation of electrical characteristics of SiC based complementary JBS structures. 2005.
24. **Rainer Taniloo**. Ökonoomsete negatiivse diferentsiaaltakistusega astmete ja elementide disainimine ja optimeerimine. 2005.
25. **Pauli Lallo**. Adaptive secure data transmission method for OSI level I. 2005.
26. **Deniss Kumlander**. Some practical algorithms to solve the maximum clique problem. 2005.
27. **Tarmo Veskioja**. Stable marriage problem and college admission. 2005.
28. **Elena Fomina**. Low power finite state machine synthesis. 2005.
29. **Eero Ivask**. Digital test in WEB-based environment 2006.
30. **Виктор Войтович**. Разработка технологий выращивания из жидкой фазы эпитаксиальных структур арсенида галлия с высоковольтным p-n переходом и изготовления диодов на их основе. 2006.
31. **Tanel Alumäe**. Methods for Estonian large vocabulary speech recognition. 2006.
32. **Erki Eessaar**. Relational and object-relational database management systems as platforms for managing softwareengineering artefacts. 2006.
33. **Rauno Gordon**. Modelling of cardiac dynamics and intracardiac bioimpedance. 2007.
34. **Madis Listak**. A task-oriented design of a biologically inspired underwater robot. 2007.