THESIS ON INFORMATICS AND SYSTEM ENGINEERING C71

Self-Diagnosis in Digital Systems

SERGEI KOSTIN

TUT
PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Engineering

This dissertation was accepted for the defence of the degree of Doctor of Philosophy in Computer and Systems Engineering on February 14, 2012.

Supervisor: Prof. Raimund-Johannes Ubar
Department of Computer Engineering, TUT

Opponents: Prof. Einar Aas
Norwegian University of Science and Technology, Norway

Prof. Vladimir Hahanov
Kharkov National University of Radioelectronics, Ukraine

Defence of the thesis: March 28, 2012

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for any academic degree.

/Sergei Kostin/

Copyright: Sergei Kostin, 2012
ISSN 1406-4731
ISBN 978-9949-23-250-5 (publication)
ISBN 978-9949-23-251-2 (PDF)

INFORMAATIKA JA SÜSTEEMITEHNIKA C71

Isediagnoosivad digitaalsüsteemid

SERGEI KOSTIN

TTÜ
KIRJASTUS

To my family

Abstract

This thesis addresses issues in the field of combinational logic diagnosis. The presented work is focused on accelerating diagnostic procedure and improving accuracy of identifying possible defect locations in digital systems that employ *built-in self-test* (BIST) facilities. Embedded diagnosis mechanisms inherent in *very-large-scale integration* (VLSI) devices have become extremely important in ensuring reasonable quality and reliability as well as enabling to perform accurate diagnosis within feasible time in these constantly increasing in complexity devices. Continuous decreasing dimensions of the transistors and interconnecting wires followed by more sophisticated fabrication techniques of manufacturing *integrated circuits* (ICs) has expanded the variety of possible defects causing a circuit to fail. To address the difficulties of handling these newly introduced defects, a fault model free approach to fault diagnosis has been proposed in the thesis.

The main contributions of the thesis are: new methods for optimized fault diagnosis applicable in BIST environments, and hierarchical approach of fault diagnosis in combinational networks based on the fault model free diagnosis concept.

A novel bisection procedure for fault location is developed in which the diagnostic weight of test patterns is taken into account. Another novelty is the sequential nature of the procedure which allows pruning the search space and also to stop when applicable diagnostic accuracy is achieved. Moreover, methods for partitioning a single signature analyzer into a set of multiple signature analyzers to improve the diagnostic resolution are developed. The ultimate goal of the proposed techniques is to optimize diagnostic procedure by minimizing diagnosis application time at the accepted diagnostic resolution.

The proposed hierarchical approach for fault diagnosis combines two opposite and conflicting trends: high-level modeling and defect-orientation. The new approach integrates cause-effect and effect-cause reasoning techniques that help to cope with the growing complexities of digital circuits and to diagnose a failing circuit accurately without using fault models.

As a result of the research, a multifunctional e-learning environment to support CAD master courses for learning, getting hands-on experience, and carrying out laboratory research in developing optimized procedures for locating faults in complex electronic systems has been developed.

Kokkuvõte

Käesolev väitekiri on pühendatud digitaalsüsteemide diagnoosi probleemide uurimisele. Töö on fokusseeritud diagnoosi kiiruse tõstmisele ja täpsuse parandamisele defektide lokaliseerimisel isetestivates digitaalsüsteemides. Süsteemide sisse ehitatud diagnoosivahendid on muutunud äärmiselt oluliseks komponentideks süsteemide töökindluse ja usaldatavuse tagamisel, mille puhul põhikriteeriumiks on rikete avastamine ja diagnoosimine võimalikult täpselt ja kiiresti üha kasvava süsteemide keerukuse tingimustes. Transistorskeemide ja nendevaheliste ühenduste dimensioonide pidev kahanemine ning kiipide valmistamistehnoloogiate üha keerukamaks muutumine on kaasa toonud füüsikaliste defektide põhjuste ja iseloomu mitmekesisuse kasvu, mis omakorda teeb üha raskemaks adekvaatselt defekte ja rikkeid modelleerida. Nimetatud põhjusel ja tehnoloogia poolt püstitatud väljakutsega hakkama saamiseks on käesolevas töös rakendatud uut paradigmat – täpset rikkemudelit mitte vajavat rikete diagnoosi.

Väitekirja põhitulemusteks on: uued meetodid rikete diagnoosi optimeerimiseks isetestivates süsteemides ja hierarhiline rikete diagnoosi kontseptsioon, mis ei vaja rikete mudelite kasutamist.

Töötati välja uudne testide bisektsioneerimismeetod rikete diagnoosimiseks, mis põhineb rikete „kaalude“ mõiste sisseviimisel ja kasutamisel. Uut tüüpi sekventsiaalse diagnoosiprotseduuri põhimõte võimaldab efektiivselt juhtida diagnostikaprotsesse, saavutamaks parimat diagnoositäpsust etteantud ajaliste piirangute juures. Diagnoosi resolutsiooni tõstmiseks on välja töötatud mitme signatuuranalüsaatori kasutamisel põhinev aparatuurne meetod ja algoritmiline lahendus. Uudsete lahenduste üldeesmärgiks on tagada võimalus optimeerida diagnostikaprotseduure, kus kriteeriumiteks on ühelt poolt diagnoosiks kulutatav aeg ja teiselt poolt diagnoosi täpsus.

Uus hierarhiline diagnoosikontseptsioon ühendab kaks teineteisele vasturääkivat trendi: kõrgtasemel modelleerimist ja defektidele orienteeritust. See põhineb kahe klassikalise strateegia – põhjus-tagajärg ja tagajärg-põhjus integreerimisel, mis võimaldab ületada keerukusest tingitud raskusi ja realiseerida veamudeli vaba diagnoosi põhimõtet.

Töö käigus loodud prototüüplahenduste baasil on rajatud e-õppe keskkond digitaaldisaini kursuste toetamiseks ja eksperimenteerimisvõimaluste andmiseks tudengitele keerukate süsteemide rikete diagnoosimisel ja kõrvaldamisel.

Acknowledgements

I would like to appreciate everyone who has supported and advised me during my Ph.D. studies.

First of all, I would like to express sincere gratitude to my supervisor Prof. Raimund-Johannes Ubar for bringing me to the world of science and research. Thanks to his comprehensive guidance and great inspiration during my Ph.D. studies I have been encouraged to complete this thesis. I am also grateful to Dr. Jaan Raik for giving valuable advices and comments during the research work behind this thesis.

Special thanks go to Dr. Margus Kruus, the head of the Department of Computer Engineering for creating favorable conditions for productive work and for his readiness to help with any administrative issues.

Furthermore, I would like to thank Anton Tsertov, Anton Tsepurov and Igor Aleksejev for being together in these eventful student years and for creating friendly working atmosphere in the room IT-231. I would like also to express my appreciation to Dr. Artur Jutman, Dr. Sergei Devadze, Dr. Maksim Jenihhin and all other colleagues from the Department of Computer Engineering not mentioned here.

Moreover, I would like to acknowledge the organizations that have supported my Ph.D. studies: Tallinn University of Technology, Centre of Research Excellence in Dependable Embedded Systems (CREDES), Centre for Integrated Electronic Systems and Biomedical Engineering (CEBE), National Graduate School in Information and Communication Technologies (IKTDK) and Estonian IT Foundation (EITSA).

Finally, I would like to express my gratitude to my family and especially to my mother who was supporting and motivating me all these years.

Thank you all!

Sergei Kostin
Tallinn, January 2012

List of Publications

Optimized fault diagnosis in the BIST environment

- R. Ubar, S. Kostin, J. Raik, ”Embedded fault diagnosis in digital systems with BIST”. *Embedded Hardware Design (Microprocessors and Microsystems)*, 2008, 32(5 - 6), pp. 279 - 287.
- R. Ubar, S. Kostin, J. Raik, ”Embedded Diagnosis in Digital Systems”, *Proc. of 26th International Conference on Microelectronics (MIEL' 2008),* Nis, Serbia, May, 2008, pp. 421 - 424.
- R. Ubar, S. Kostin, J. Raik, “Built-In Self Diagnosis with Multiple Signature Analyzers in Digital Systems”, *Proc. of 9th IEEE Latin American Test Workshop (LATW' 2008)*, Puebla, Mexico, February, 2008, pp. 29 - 34.
- S. Kostin, “Fault Diagnosis in the BIST Environment Based on Bisection of Detected Faults”, M. Sc. thesis, Tallinn University of Technology, 2007
- R. Ubar, S. Kostin, J. Raik, M. Kruus, “Experimental Comparison of Different Diagnosis Algorithms in the BIST Environment”. *IASTED Conference on Applied Simulation and Modelling (ASM' 2007).*
- R. Ubar, S. Kostin, J. Raik, “Experimental Comparison of Different Diagnosis Algorithms in the BIST Environment”, *Proc. of 8th IEEE Latin-American Test Workshop (LATW' 2007)* (1 - 6). IEEE Computer Society
- R. Ubar, S. Kostin, J. Raik, T. Evartson, H. Lensen, “Fault Diagnosis in Integrated Circuits with BIST”, *Proc. of 10th EUROMICRO Conference on Digital System Design (DSD' 2007),* IEEE Computer Society Press, pp. 604 - 610

Fault model free fault diagnosis

- R. Ubar, S. Kostin, J. Raik, “Investigations of the Diagnosibility of Digital Networks with BIST”, *Proc. of 10th IEEE Latin-American Test Workshop (LATW' 2009)* (1 - 6), Rio de Janeiro, Brazil, 2009

- R. Ubar, S. Kostin, J. Raik, “Block-Level Model-Free Debug and Fault Diagnosis in Digital Systems”, *Proc. of 12th EUROMICRO Conference on Digital System Design (DSD’ 2009),* Patras, Greece, August 27 - 29, 2009, pp. 229-232.
- R. Ubar, S. Kostin, J. Raik, “Combined Fault-Model Free Cause-Effect and Effect-Cause Fault Diagnosis in Block-Level Digital Networks”, *Proc. of 1st Asia Symposium on Quality Electronic Design (ASQED’09)*, Kuala Lumpur, Malaysia, July 15 - 16, 2009, pp. 385 - 390.
- R. Ubar, S. Kostin, J. Raik, “Calculation of the Diagnosibility of Digital Circuits without Using Fault Models”, *Proc. of the 11th Biennial Baltic Electronic Conference (BEC’ 2008),* Tallinn, Estonia, October 6-8, 2008, pp. 159-162

Hierarchical fault diagnosis

- S. Kostin, “Macro Level Defect-Oriented Diagnosability of Digital Circuits”, *PhD forum at Conference Design, Automation and Test in Europe (DATE’ 2011)*, Grenoble, France, 2011
- S. Kostin, R. Ubar, J. Raik, M. Brik, “Hierarchical physical defect reasoning in digital circuits”, *Estonian Journal of Engineering*, 2011, 17(3), pp. 1-15
- R. Ubar, S. Kostin, J. Raik, “Defect-Oriented Module-Level Fault Diagnosis in Digital Circuits”, *Proc. of 14th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS’ 2011)*, Cottbus, Germany, April 13 - 15, 2011, pp. 81 - 86.
- S. Kostin, R. Ubar, J. Raik, “Macro Level Defect-Oriented Diagnosability of Digital Circuits”, *Proc. of the 12th Biennial Baltic Electronic Conference (BEC’ 2010)*, Tallinn, Estonia, October 4-6, 2010, pp. 149 - 152.

Laboratory environment for education and research of test and fault diagnosis

- S. Kostin, A. Tsertov, A. Tsepurov, I. Aleksejev, R. Ubar, A. Jutman, J. Raik, S. Devadze, “BIST Analyzer and Diagnozer”, *In University Booth section of Design, Automation and Test in Europe (DATE’ 2010)*, Dresden, Germany, March, 2010.
- R. Ubar, A. Jutman, J. Raik, S. Devadze, I. Aleksejev, A. Chepurov, A. Chertov, S. Kostin, E. Orasson, H.-D. Wuttke, “E-Learning Environment for WEB-Based Study of Testing”, *Proc. of the 8th European Workshop on Microelectronics Education (EWME’ 2010)*, Germany, Darmstadt, 2010, pp. 47 - 52.

- R. Ubar, S. Kostin, A. Jutman, J. Raik, H.-D. Wuttke, “DIAGNOZER: A Laboratory Tool for Teaching Research in Diagnosis of Electronic Systems”, *Proc. of International Conference on Microelectronic Systems Education*, San Francisco, USA, July 25-26, 2009, pp. 12 - 15.
- S. Kostin, R. Ubar, J. Raik, M. Aarna, M. Brik, H.-D. Wuttke, “Teaching Research in the Laboratory Using Diagnosis Environment for Digital Systems”, *Proc. of 20th EAEEIE Annual Conference on Innovation in Education for Electrical and Information Engineering*, Valencia, Spain, June 22-24, 2009, 1 - 4.

List of Abbreviations

ATE	Automatic Test Equipment
ATPG	Automatic Test Pattern Generator
BIST	Built-In Self-Test
BISTA	BIST Analyzer
BISD	Built-In Self-Diagnosis
CAD	Computer-Aided Design
CMOS	Complementary Metal-Oxide-Semiconductor
CUT	Circuit Under Test
CUD	Circuit Under Diagnosis
DFR	Design For Reliability
DFT	Design For Testability
DM	Diagnostic Matrix
DP	Diagnostic Point
DPM	Defects Per Million
DT	Diagnostic Tree
EDA	Electronic Design Automation
EDIF	Electronic Design Interchange Format
FFR	Fanout-Free Region
HD	Hamming Distance
HDL	Hardware-Description Language
HTTF	Hard-To-Test Faults
IC	Integrated Circuit

I/O	Input/Output
LFSR	Linear Feedback Shift Register
LSI	Large-Scale Integration
LSSD	Level-Sensitive Scan Design
MISR	Multiple Input Signature Register
ORA	Output Response Analyzer
PCB	Printed Circuit Board
PI	Primary Input
PO	Primary Output
PPM	Parts Per Million
PRPG	Pseudo-Random Pattern Generator
RTL	Register-Transfer Level
SA	Signature Analyzer
SA0	Stuck-At-0
SA1	Stuck-At-1
SAF	Stuck-At Fault
SSF	Single Stuck-At Fault
SSBDD	Structurally Synthesized Binary Decision Diagram
TPG	Test Pattern Generator
VHDL	VHSIC Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuits
VLSI	Very-Large-Scale Integration
XOR	Exclusive OR

Contents

CHAPTER 1 INTRODUCTION...23
1.1 Motivation 23
1.2 Problem formulation...24
1.3 Thesis contribution 25
1.4 Thesis structure..26
CHAPTER 2 BACKGROUND 29
2.1 Introduction to digital test and diagnosis..29
2.2 Approaches to digital logic testing 31
2.3 Logical fault models 33
2.3.1 Stuck-at fault model 34
2.4 Design for testability 35
2.5 Logic built-in self-test 37
2.5.1 General architecture of BIST...38
2.5.2 Linear Feedback Shift Register 39
2.5.3 LFSRs used as pseudorandom test pattern generators................41
2.5.4 LFSRs used as signature analyzers..42
2.6 Logic diagnosis..44
2.6.1 Cause-effect analysis 45
2.6.2 Effect-cause analysis 47
2.7 State-of-the-art...50
2.7.1 BIST environment challenges for fault diagnosis 50
2.7.2 The strategies for diagnosis 51
2.7.3 Fault models 52
2.7.4 Fault simulation 53
2.7.5 A hierarchical approach to fault diagnosis 53
2.7.6 E-learning in digital fault diagnosis..54
2.7.7 Conclusions 55
CHAPTER 3 FAULT DIAGNOSIS IN THE BIST ENVIRONMENT.....57
3.1 Search procedures for diagnosis 57
3.1.1 Binary Search 58

3.1.2 Digging ... 59
3.1.3 Doubling and Jumping ... 59
3.1.4 Summary ... 60
3.2 Diagnosis with bisection by fault coverage ... 60
3.2.1 Built-in self-diagnosis environment ... 60
3.2.2 Algorithm description ... 61
3.2.3 Example of bisection detected faults ... 63
3.2.3.1 Primary diagnostic tree ... 64
3.2.3.2 Secondary diagnostic tree ... 66
3.2.4 Logical operations needed to support the proposed method ... 68
3.2.4.1 Calculation of total fault vectors ... 68
3.2.4.2 Updating list of suspected faults ... 69
3.2.5 A modification to implementation of the proposed method in BIST environment ... 70
3.3 Experimental data ... 72
3.3.1 Analysis of results obtained during comparison between the initial method and the modified method ... 72
3.3.2 Comparison of searching algorithms ... 74
3.4 Conclusions ... 76
CHAPTER 4 METHODS FOR INCREASING DIAGNOSTIC RESOLUTION ... 77
4.1 Fault diagnosis challenges in BIST environment ... 77
4.2 BIST with multiple signature analyzers ... 78
4.3 Design of the interface between CUT and SAs ... 80
4.3.1 Algorithm “Equal Subsets” ... 80
4.3.2 Algorithm “Unique Faults” ... 83
4.4 Fault diagnosis with a set of SAs ... 85
4.5 Experimental data ... 86
4.5.1 Algorithms of designing the interface between CUT and SAs ... 86
4.5.2 Comparison of different fault diagnosis methods ... 88
4.5.3 Trade-off between time cost and accuracy of the fault diagnosis ... 90
4.6 Conclusions ... 93
CHAPTER 5 FAULT MODEL FREE HIGH-LEVEL FAULT DIAGNOSIS ... 95
5.1 Block-level fault-model free diagnosis ... 95
5.2 Diagnostic matrix of a network ... 96
5.3 Lower bound of average block-level diagnosability ... 99
5.4 Diagnosis with adjusting the test responses ... 100

5.5 Probabilistic diagnosability measure 102
5.6 Improving the diagnosability 103
5.7 Experimental data 105
5.8 Conclusions 108
CHAPTER 6 HIERARCHICAL FAULT DIAGNOSIS 111
6.1 General description of the method 111
6.2 Cause-effect high-level fault diagnosis 113
6.3 Defect reasoning in modules by using the conditional stuck-at fault model 115
6.4 Effect-cause high-level fault reasoning 117
6.5 Low-level fault diagnosis 119
6.6 Experimental data 120
6.7 Conclusions 123
CHAPTER 7 IMPLEMENTATION OF THE RESEARCH RESULTS IN THE TEACHING ENVIRONMENT 125
7.1 Overview of fault diagnosis problems 125
7.2 Description of the environment 126
7.3 Laboratory research scenarios 129
7.3.1 Research Scenario 1: Diagnostic Circuitry 130
7.3.2 Research Scenario 2: Diagnostic Algorithms 132
7.3.3 Research Scenario 3: Object of Diagnosis 135
7.4 Conclusions 137
CHAPTER 8 CONCLUSIONS 139
8.1 Main results 139
8.2 Future work 141
REFERENCES 143

Chapter 1
INTRODUCTION

This introductory chapter gives an overview of the area addressed by current thesis. At first, the motivation for the work is given followed by the formulation of the problem and the outline of main contributions. The last part of the chapter describes the organization of the thesis.

1.1 Motivation

The impact of the semiconductor *integrated circuit* (IC) on modern life has been so profound that it is now often taken for granted. Nowadays, we entrust heavily our lives to electronic appliances and we are being assured in the reliability and safety of these devices. Furthermore, we have come to expect increasingly sophisticated electronics products at ever lower prices, whereas the business world has come to expect greater productivity through improved information technology [1].

The IC industry’s desire to reduce chip area in order to realize higher yields and more circuits sites per wafer has been the primary motivation for transistor scaling over last decades. That is why Moore’s law continues to drive the scaling of *complementary metal-oxide-semiconductor* (CMOS) technology [2]. The feature size of the transistor now has been shrunk well into nanoscale region [3]. Today a single *very-large-scale integration* (VLSI) device can contain over one billion transistors. The ever-increasing level of integration has enabled higher performance and richer feature sets on a single chip. However, continuous transistor miniaturization requires more advanced design techniques to cope with the increased process variation, interconnect processing difficulties, and other newly

emerging physical effects [4] in order to assure reasonable quality and reliability for future products of microelectronics.

During the IC design and manufacturing cycle, a manufacturing test screens out the bad chips. Fault diagnosis is used to find out why the bad chips have failed, which is especially important when the yield is low or when a customer returns a failed chip. Understanding how integrated circuits fail helps to identify and eliminate the causes of failures. Unlike board-level diagnosis, the main objective in diagnosing ICs is to understand the failures (to specify the fault or erroneous state) and to prevent them from recurring, not to repair the failures. Locating faults in chips and analyzing failure trends can lead to corrective design rule changes. Failure trends also help to reveal process and manufacturing problems.

Diagnosis can identify reliability trouble spots in a design and lead to special *design for reliability* (DFR) actions, and in such a way it can have an impact on future designs. Diagnosing customer rejects often results in an improved test program, which reduces *defects per million* (DPM) [5].

1.2 Problem formulation

The need for debug/diagnosis capability is essential since it usually occurs during three phases in the lifetime of a chip. After the design is fabricated on silicon for first time, popularly known as the "first silicon", a substantial amount of effort is put on debugging the first silicon. Such debug efforts usually weed out problems in the design, including design errors and design marginality.

Due to the growing intricacy of the manufacturing process, device complexity, and random variations in the manufacturing process, all fabricated chips do not meet targeted specifications. The integrated circuit manufacturers need debug/diagnosis capability to identify any persistent anomaly in the manufacturing process to achieve better yield and reliability of their products.

Field diagnosis is the last phase where diagnosis capability is required. Such field diagnosis can provide valuable information about the reliability of the device and might point out possible weaknesses in the design and/or production process for that device [6].

Development and manufacturing failures are unfortunately an inherent part of the microelectronics. The impact of such failures, whether anticipated or sudden, ranges from consequential to catastrophic. Because narrow market opportunities often drive shortened product cycles, there is need to understand failures and take corrective actions quickly. Electrical characterization, statistical analysis, signature analysis, and process experiments can provide important clues that allow inferring the cause of failure. But only full root-cause physical failure analysis can provide the incriminating evidence necessary to correct problems with confidence. When rapid corrective actions can be accomplished confidently, yield and reliability

learning, time-to-market, and end-customer satisfaction improvements are direct benefits of the process.

The crucial element of failure analysis is fault localization, a task for which both hardware and software techniques exist. Trends toward denser circuits and more sophisticated packaging, however, are limiting physical access to internal chip circuitry and, thus, decreasing the effectiveness of hardware-based diagnostics. Therefore, efficient software diagnostics require an aggressive *design-for-test* and diagnostic strategy. However, such a strategy is threatened by the microelectronics industry's strong focus on cost reduction in the design and test phases of product development [7]. Therefore, new approaches for efficient, accurate and low-cost failure analysis are still emerging.

1.3 Thesis contribution

The presented work is focused on accelerating diagnostic procedures and improving accuracy of identifying possible defect locations in digital systems that employ *built-in self-test* (BIST) facilities. To overcome the difficulties of handling continuously growing variability and complexity of physical defects in today's nanoelectronics technologies, a fault model free approach to fault diagnosis is proposed and elaborated in the thesis.

The main contributions of the current thesis can be divided into two groups: (1) optimization of fault diagnosis in digital circuits with BIST, and (2) fault model free hierarchical approach to fault diagnosis.

(1) Optimization of fault diagnosis:

- A novel optimized fault diagnosis procedure for *built-in self-test* (BIST) environments has been developed. Instead of the classical bisectioning of patterns in pseudorandom test sequences by simply counting them, a **novel bisectioning procedure** which takes into account the diagnostic information inherent in test patterns is proposed.

- Another novelty is the **sequential approach to diagnosis** which allows pruning the search space. Opposite to the classical approach which targets all failing patterns, in the proposed method not all failing patterns are necessarily needed to be fixed for diagnosis. This allows to make a trade-off between the speed of diagnosis and diagnostic resolution. The superiority of the new method is demonstrated by comparison with three known fault diagnosis methods: *classical Binary Search*, *Doubling* and *Jumping*.

- To improve the accuracy of fault diagnosis in digital systems with BIST facilities, a new concept of test response processing with **multiple signature analyzers** is proposed. The concept is based on partitioning of a single signature analyzer into a set of multiple independent analyzers, and

on optimization of the interface between the circuit under test and the analyzer block.

(2) Fault model free hierarchical fault diagnosis:

- A conception for diagnosis of digital circuits which does not use fault models is introduced. The conception allows to cope with the complexity of circuits, with the multiplicity of faults and with the problems of continuously increasing variety of physical defects in today's nanoelectronics circuits.
- A novel **hierarchical approach** for fault diagnosis in digital circuits has been developed. To cope with the complexity of problem, two opposite and conflicting trends, high-level modeling and defect-orientation are integrated. The key solutions are cause-effect analysis using **high-level faulty module dictionary**, combined with effect-cause diagnosis based on the **fault model free error reasoning**.
- For calculating the diagnosability of the given circuit, a **novel fault model free measure of diagnosability** has been elaborated which can be used for redesign of the circuit to improve the exactness of locating the faults or faulty regions in digital circuits.

As a result of the research, a multifunctional e-learning environment to support CAD master courses for learning, getting hands-on experience, and carrying out laboratory research in developing optimized procedures for locating faults in complex electronic systems has been developed. It is a combination of software tools to simulate a system under diagnosis, to emulate a pool of different strategies, methods, and algorithms of diagnostic reasoning and fault location, and to experiment with different embedded self-diagnosing architectures. Hands-on experiments target research teaching issues. The interactive character of the environment makes experiments attractive and helps to raise the students' curiosity.

1.4 Thesis structure

The presented thesis is organized into 8 chapters.

Chapter 2 provides background information on digital test and fault diagnosis, and makes a review of the state-of-the-art in the area addressed by the thesis.

Chapter 3 presents an optimized fault diagnosing procedure applicable in *built-in self-test* (BIST) environments and compares it with other known fault diagnosis methods.

Chapter 4 presents a method to improve fault diagnosis accuracy in digital systems using BIST facilities. The diagnostic resolution improvement is attained by partitioning a single signature analyzer into a set of multiple independent

analyzers. The algorithms are given to synthesize an optimal interface between the outputs of the circuit under test and the signature analyzers.

Chapter 5 introduces a concept of fault model free diagnosis combined with cause-effect analysis in digital systems represented as networks of functional blocks. A measure is proposed for evaluating the block-level diagnosability of a given network which can be used for redesign of the circuit to improve the exactness of locating the faults regions in digital circuits.

Chapter 6 presents a hierarchical approach for fault diagnosis in combinational digital circuits represented as a network of modules. The new approach integrates cause-effect and effect-cause reasoning techniques that help to cope with the growing complexities of digital circuits and to diagnose a failing circuit accurately without using fault models.

Chapter 7 presents a multifunctional e-learning environment with remote access for learning, getting hands-on experience, and carrying out laboratory research in developing optimized procedures for locating faults in complex electronic systems.

Chapter 8 draws conclusions for this thesis and discusses possible perspectives for future work.

Chapter 2
BACKGROUND

This chapter presents background information on the topics related to current research. The chapter begins with the brief introduction to digital test and fault diagnosis followed by the review of the approaches to digital logic testing. The stuck-at fault model is described since it mostly used for fault diagnosis by methods proposed in the thesis. Then the brief overview of *design for testability* (DFT), *built-in self-test* (BIST) and fault analysis techniques is presented, and finally, the review of the state-of-the-art in this area is given.

2.1 Introduction to digital test and diagnosis

The introduction of *integrated circuits* (ICs), commonly referred to as microchips or simply chips, was followed by the need to test these devices. Testing of devices that contained only hundreds of transistors was relatively simple. However, in the 1970s, *large-scale integration* (LSI) devices with thousands and tens of thousands of transistors complicated testing considerably [8]. Nowadays, *very-large-scale integration* (VLSI) devices with many millions of transistors are widely spread in common computers and electronic appliances. This is a direct result of the continuous decreasing dimensions of the transistors and interconnecting wires from micron- to nanoscale. The reduction in feature size has also resulted in increased operating frequencies and clock speeds, but introduces many new testing challenges.

Recent advances in physics, chemistry, and materials science have allowed production of nanoscale structures using sophisticated fabrication techniques. It is well known that nanoscale devices have much higher manufacturing defect rates

compared to conventional *complementary metal-oxide semiconductor* (CMOS) devices [8]. Nanoscale devices have much lower current drive capabilities and are much more sensitive to noise-induced errors such as *crosstalk*. These devices are more susceptible to failures of transistors and wires due to soft (cosmic) errors, process variations, electromigration, and material aging. As the integration scale increases, more transistors can be fabricated on a single chip, thus reducing the cost per transistor. However, the difficulty of testing each transistor grows due to the increased complexity of the VLSI device and increased potential for defects, as well as the difficulty of detecting the faults produced by those defects.

Continuous increase in size and complexity of circuits on a chip, often with little or no increase in the number of I/O (*input*/*output*) pins, produces a testing bottleneck [9]. Much more logic must be controlled and observed while the number of I/O pins remains the same that makes testing of chip a real challenge. Moreover, the need for testing continues to grow in importance since just one single faulty transistor or wire can make the entire chip fail to function properly or at the required operating frequency. Furthermore, manufacturing defects are unavoidable in producing nanoscale devices. As a result, a number of ICs are expected to be faulty and, therefore, testing is required to guarantee fault-free products. It is also necessary to test components of VLSI devices at various stages during the manufacturing process. For example, in order to produce an electronic system, first, ICs must be produced and used to assemble *printed circuit boards* (PCBs), and then the PCBs are used to assemble the system. The cost of detecting a faulty IC increases by an order of magnitude when moving through each stage of manufacturing, from device level to board level to system level and finally to system operation in the field [8]. Random defects in individual components may not have significant impact on a product success, but a defective manufacturing process for a complex VLSI device, or a design error in some obscure function, could escape detection until first product returns that may result a very expensive product recall [9].

Due to advances in semiconductor technologies new *electronic design automation* (EDA) tools allow to design and fabricate chips of greater complexity at lower cost. As a result, testing consumes a greater percentage of total production cost. It requires also more effort to create a test program as well as more stimuli to exercise the chip. Both, the cost of developing tests and the cost of applying test to individual units must be considered. In some cases, it becomes necessary to invest more effort into initially creating a test in order to reduce the cost of applying it to individual units. The difficulty in creating test programs for new designs also contributes to delays in getting products to the market. Therefore, new test strategies are emerging in response to test problems arising from these increasingly complex devices, and greater emphasis is placed on finding defects as early as possible in the manufacturing cycle. New algorithms are being devised to create tests for logic circuits, and more attention is being given to *design for testability*

(DFT) techniques that require participations by logic designers, who are being asked to adhere to design rules that facilitate design of more testable circuits [9].

2.2 Approaches to digital logic testing

Testing typically consists of applying a set of test stimuli to the inputs of the *circuit under test* (CUT) while analyzing the output responses [8]. The stimuli are called *test patterns* or *test vectors*. Each bit of the vector is applied to a specific input pin of the CUT. The expected or predicted outcome is usually observed at output pins of the device, although some test configurations permit monitoring of *test points* within the circuit that are not normally accessible during operation. A tester captures the response at the output pins and compares that response to the expected response determined by applying the stimuli to a known good device and recording the response, or by creating a *model* of the circuit (i.e. a representation or abstraction of the selected features of the system) and simulating the input stimuli by means of that model [9].

Circuits that produce the correct output responses for all input stimuli pass the test and are considered to be *fault-free*. If the CUT response differs from the expected response at any point during the test sequence, then an *error* is said to have occurred. The error results from a *defect* in the circuit and this CUT is assumed to be *faulty*. A *fault* is a representation of defect reflecting a physical condition that causes a circuit to fail to perform correctly. A *failure* is a deviation in the performance of a circuit or system from its specified behavior and represents an irreversible state of a component. A circuit *defect* may lead to a *fault*, a *fault* can cause a circuit *error*, and a circuit *error* can result in a system *failure* [8].

Generation of effective test patterns is an important part of the test. The patterns (input stimuli) can be created in the following ways:

- generate all possible combinations of input stimuli
- develop test sequences targeted the functionality of the design (functional testing)
- create test sequences targeted at specific faults (structural testing)
- generate pseudorandom sequences

Early approaches to test generation involved the application of all possible binary combinations to device inputs to perform a complete functional verification of the device. This approach is referred to as *exhaustive testing*. Application of 2^n test vectors to a device with n inputs was effective when n was small and there were no sequential circuits on the board. Since the number of tests grows exponentially with n, this approach quickly became infeasible due to prohibitively huge number of patterns to be applied.

Another approach consists in testing the functionality of a circuit by generating sequences of input stimuli intended to drive the circuit through all different internal

states, while varying the conditions on the data-flow inputs. This approach is referred to as *functional testing*. If the circuit responds correctly to all applied input patterns, it can be concluded that the circuit is defect free. However, it is incorrect conclusion since it may occur that one or more defects presented in the circuit were not detected by the applied stimuli. Thus, the major problem with this approach is an inability to evaluate the effectiveness of the test stimuli. Effectiveness can be estimated by observing the number of defective products returned by the customer, so-called "*test escapes*", but this is a costly solution.

In 1959, R.D. Eldred suggested to test hardware rather than function [9]. This proposed approach relied on generating tests for specific faults. The most commonly occurring faults would be modeled and input stimuli created to test for the presence or absence of each of these faults. Thus, this approach that employed circuit structural information and a set of *fault models* to generate test vectors was referred to as *structural testing*.

Structural testing saves test application time and improves test efficiency, as the total number of test patterns applied is decreased, because the test vectors target only specific faults that would result from defects in the manufactured circuit. On the other hand, structural testing cannot guarantee detection of all possible manufacturing defects, as the test vectors are generated based on specific fault models. However, the use of fault models does provide a quantitative measure of the fault-detection capabilities of a given set of test vectors for a targeted fault model. The *quality*, or the effectiveness, of a test is measured by the ratio between the number of faults it detects and the total number of faults in the assumed fault universe; the ratio is referred to as the *fault coverage*.

It may be impossible to obtain 100% fault coverage because of the existence of *undetectable faults*. An *undetectable fault* means there is no test to distinguish the fault-free circuit from a faulty circuit containing that fault. As a result, the *test evaluation*, i.e. fault coverage, can be modified and expressed as the *fault detection efficiency*, also referred to as the *effective fault coverage*, which is defined as ratio between the number of detected faults by given test and the number of *detectable* faults [8]. In order to calculate *fault detection efficiency*, all undetectable faults in the circuit must be correctly identified, which is usually a difficult task.

Fault coverage is related to the yield and the *defect level* by the following expression [10]:

$$\text{Defect level} = 1 - \text{yield}^{\,(1 - \text{fault coverage})}$$

From this equation results that both, yield and fault coverage, have direct impact on a reject rate of total number of manufactured devices. Nevertheless, improving fault coverage can be easier and less expensive than improving manufacturing yield because making yield enhancements can be costly. Therefore, generating test patterns with high fault coverage is very important. Of course, if there is a yield crash, i.e. a sudden, significant drop in the number of devices that pass a test,

diagnosis must be performed to identify the possible causes. To aid investigations, it may be necessary to generate additional test vectors specifically for the purpose of isolating the source of the crash. Yield problems commonly arise in the early stages of the production, when manufacturing processes are new and unfamiliar to employees [9]. As a result, there are likely to be more occasions when it is necessary to investigate problems in order to diagnose causes. For mature products yield is frequently quite high, and testing may consist of sampling by randomly selecting parts for test.

If the CUT responds correctly to all applied stimuli, confidence in the CUT increases. However, it cannot be concluded that the circuit is fault-free. It can be only concluded that the CUT does not contain any of the faults for which it was tested, but it could contain other faults for which an effective test was not applied. Therefore, the goal of *test generation* is to find an efficient set of test vectors that detects all faults considered for that circuit. Because a given set of test vectors is usually capable of detecting many faults in a circuit, *fault simulation* is typically used to evaluate the *fault coverage* obtained by that set of test vectors. As a result, fault models are needed for fault simulation as well as for test generation [8].

2.3 Logical fault models

The diversity of possible physical defects inside VLSI device makes generation of test patterns extremely difficult. Modeling physical defects as logical faults, which represent the effect of physical defects on the behavior of the modeled system, helps to facilitate the problem of fault analysis, since it becomes a logical rather a physical problem. Thus, the complexity of the problem is reduced considerably since many different physical defects may be modeled by a single logical fault.

Fault models are necessary mainly for generating and evaluating a set of test vectors. Typically, a good fault model should satisfy two criteria: (1) it should accurately reflect the behavior of defects, and (2) it should be computationally efficient in terms of fault simulation and test pattern generation. Many fault models have been proposed [8]: stuck-at fault, switch-level fault (transistor stuck-open or stuck-short), delay fault (gate-delay, transition, path-delay), parametric fault, bridging fault (wired-AND/wired-OR, dominant, dominant-AND/dominant-OR), crosstalk effects defining fault models, etc. Unfortunately, no single fault model accurately reflects the behavior of all possible defects that can occur. As a result, a combination of different fault models is often used in the generation and evaluation of test vectors and testing approaches developed for VLSI devices.

Further, more detailed overview of stuck-at fault model is presented, since it has been widely employed in the experimental part of the thesis for fault injection, fault simulation and fault diagnosis with the goal to prove efficiency of the proposed fault diagnosis methods.

2.3.1 Stuck-at fault model

The *single stuck-at fault* (SSF) is a logical fault model that is also referred to as the *classical* fault model because it has been the first and the most widely studied and used [11]. A stuck-at fault affects the state of logic signals on lines in a logic circuit, including *primary inputs* (PIs), *primary outputs* (POs), internal gate inputs and outputs, fanout stems (sources), and fanout branches. A stuck-at fault transforms the correct value on the faulty signal line to appear to be stuck at a constant logic value, either logic 0 or logic 1, referred to as *stuck-at-0* (SA0) or *stuck-at-1* (SA1), respectively [8]. Although stuck-at fault model validity is not universal, it has following useful attributes [11]:

- represents many different physical faults [12]
- technology-independent since it can be applied to any structural model
- experience has shown that tests that target SSFs detect also many non-classical faults
- the number of SSFs in a circuit is quite small compared to other fault models; moreover, the number of faults can be reduced by fault collapsing techniques and, therefore, simulation of SSFs is computationally very efficient
- SSFs can be used to model other types of faults

In case of *single-fault assumption*, the number of possible faults is equal to $2n$, where n – number of signal lines in the CUT. In reality, several lines can be simultaneously stuck-at that result in *multiple-fault assumption*. Thus, $3^n - 1$ various combinations of multiple stuck-at faults may occur in a CUT, where each line n can have one of three possible states: SA0, SA1 or fault-free. Of these possibilities is excluded a combination where all lines are fault-free, which corresponds to completely fault-free circuit.

When considering multiple stuck-at faults, even relatively small number of signal lines n in a CUT leads to problem of "combinatorial explosion", i.e. the number of problems to be solved explodes. Therefore, the attempt to test for every multiple stuck-at fault combination is clearly impractical and it has led to adoption of the *single-fault* assumption. When creating a test, it is assumed that a single fault exists. Most frequently, it is assumed that an input or output of a gate is SA1 or SA0. Many years of experience with stuck-at fault model by many digital electronics companies has demonstrated that it is effective [9]. A good stuck-at test which detects all or nearly all single stuck-at faults in a circuit will also detect all or nearly all multiple stuck-at faults and short faults. There are technology dependent faults for which the stuck-at fault model must be modified or augmented.

The single stuck-at fault model can also be applied to sequential circuits. However, high fault coverage test generation for sequential circuits is much more

difficult than for combinational ones, since for most faults in a sequential logic circuit is needed to generate sequences of test vectors. Therefore, DFT techniques are frequently used to ease test generation for sequential circuit [8]. Although it is physically possible for a line to be SA0 or SA1, many other defects within a circuit can also be detected with test vectors developed to detect stuck-at faults. The idea of *N-detect* single stuck-at fault test vectors was proposed to detect more defects not covered by the stuck-at fault model [13]. In *N-detect* set of test vectors, each single stuck-at fault is detected by at least *N* different test vectors. However, test vectors generated using the stuck-at fault model do not necessarily guarantee the detection of all possible defects, so other fault models are needed.

Under the single-fault assumption, two or more faults may result in identical faulty behavior for all possible input patterns. These faults are called *equivalent faults* and can be represented by any single fault from the set of equivalent faults [8]. The reduction of the entire set of single faults by removing equivalent faults is referred to as *fault collapsing*. Stuck-at fault collapsing typically reduces the total number of faults by 50 to 60% [14]. Fault collapsing for stuck-at faults is based on the fact that a SA0 at the input of an AND (NAND) gate is equivalent to the SA0 (SA1) at the output of the gate. Similarly, a SA1 at the input to an OR (NOR) gate is equivalent to the SA1 (SA0) at the output of the gate. For an inverter, a SA0 (SA1) at the input is equivalent to the SA1 (SA0) at the output of the inverter. Furthermore, a stuck-at fault at the source (output of the driving gate) of a fanout-free net is equivalent to the same stuck-at fault at the destination (gate input being driven). Thus, fault collapsing helps to reduce both test generation and fault simulation times.

2.4 Design for testability

During the early stages of IC production history design and test were performed by separate and unrelated groups of engineers since they were considered as separate tasks [8]. A design engineer implemented the required functionality based on design specifications without imaging how the manufactured chip had to be tested. Once the functionality was implemented, the design information was transferred to test engineers. A test engineer determined the efficient way of testing each manufactured device within a reasonable time, in order to screen out devices containing manufacturing defects and ship all defect-free devices to customers. The final quality of the test was determined by the number of defective devices shipped to the customers, based on customer returns, and was measured in terms of defective *parts per million* (PPM) shipped.

This approach worked well for *small-scale integrated* circuits that mainly consisted of combinational logic or simple finite-state machines. However, it became infeasible when complexity of designs had moved to *very-large-scale integration* (VLSI). During the 1980s testing of these VLSI devices was primarily

relied on fault simulation to measure the fault coverage of the supplied functional patterns. The goal of functional patterns was to navigate through the long sequential depths of a design accessing all internal states and detecting all possible manufacturing defects. A *fault simulation* tool was employed to evaluate the effectiveness of the functional patterns in terms of quantity. If the available functional patterns did not achieve the target fault coverage, additional patterns were generated and applied. However, this approach usually did not achieve the circuit's fault coverage beyond 80%, and the quality of the shipped products continued to suffer [8].

It became gradually clear that designing devices without paying much attention to test resulted in increased test cost and decreased test quality. Some designs that were best in class, concerning their functionality and performance, failed commercially due to extremely high test cost or poor product quality [8]. Thus, the cost of testing a system has become a major component in the cost of designing, manufacturing, and maintaining a system. The cost of testing reflects many factors [11] such as the cost of test pattern generation, the cost of fault simulation and generation of fault location information, the cost of test equipment and the cost of test application time. Because these costs can be high and may even exceed design costs, it is important that they are kept within reasonable bounds. Test cost limits have led to the development and deployment of DFT engineering in the industry. DFT techniques are usually divided into three categories [8]: (1) *ad hoc DFT* techniques, (2) *level-sensitive scan design* (LSSD) or *scan design*, or (3) *built-in self-test* (BIST).

The first DFT techniques introduced in the 1970s were *ad hoc* methods that targeted only those parts of the circuit that would be difficult to test and added extra circuitry to improve the *controllability* or *observability* [11]. *Controllability* means the ability to establish a specific signal value at each node (line) in a circuit by setting values on the primary inputs of a circuit. *Observability* means the ability to determine the signal value at any node (line) in a circuit by controlling the inputs and observing the outputs of a circuit. *Ad hoc* techniques typically employed test point insertion to access internal nodes directly. Substantially, *ad hoc* methods have managed to improve the testability of a design and to simplify sequential *automatic test pattern generation*. However, it was still a challenge to obtain more than 90% fault coverage for large designs [8]. The reason was that deriving functional patterns by hand or generating test patterns for a sequential circuit is a much more difficult problem compared to test patterns generation for a combinational circuit [15] [14] [16].

The next and the most important DFT technique proposed was *level-sensitive scan design*, also referred to as *scan design* [17]. In a flip-flop-based scan design, testability is improved by adding extra logic to each flip-flop in the circuit to form a shift register, or scan chain. During the *scan mode*, the scan chain is used to shift in (or scan in) a test vector to be applied to the combinational logic. During one clock cycle in the *system mode* of operation, the test vector is applied to the

combinational logic and the output responses are captured into the flip-flops. Then the circuit is switched back to *scan mode*, so as to shift out (or scan out) the combinational logic output response while shifting in the next test vector to be applied [8]. As the result, scan based design technique reduces the problem of testing sequential logic to the problem of testing combinational logic.

Built-in self-test (BIST) was introduced around 1980 [18] [19] to integrate a *test pattern generator* (TPG) and an *output response analyzer* (ORA) in the VLSI device to perform testing internal to the IC. Since the tester functions reside with the CUT, BIST can be applied at all levels of testing, from wafer through system-level testing.

2.5 Logic built-in self-test

Recent advances in semiconductor manufacturing technology have produced various testing challenges in the production and usage of VLSI circuits during wafer probe, wafer sort, pre-ship screening, incoming test of chips and boards, test of assembled boards, system test, periodic maintenance, repair test, etc. Traditional test techniques that employ ATPG software have become quite expensive and can no longer achieve optimal fault coverage for deep nanometer designs from the chip level to the board and system levels [8].

One way to overcome these testing problems is to incorporate *built-in self-test* (BIST) features into a digital circuit at the design stage [20] [11] [14] [21] [19] [22]. Logic BIST is a *design for testability* (DFT) technique, in which circuits that generate test patterns and analyze the output responses of the functional circuitry are embedded in the chip or elsewhere on the same board where the chip resides. Logic BIST is crucial for many applications, especially concerning life-critical and mission-critical ones. These applications commonly found in the automotive, aerospace/defense, telecommunications, banking, computer, healthcare and networking industries require on-chip, on-board, or in-system self-test to improve the reliability of the entire system, as well as the ability to perform remote diagnosis [8].

BIST techniques can be classified into two categories [11], namely, *on-line* BIST, which includes *concurrent* and *non-concurrent* techniques, and *off-line* BIST, which includes *functional* and *structural* approaches.

In *on-line* BIST, testing is carried out during normal operational mode; i.e. the *circuit under test* (CUT) is not placed into a test mode where normal functional operation is locked out. *Concurrent on-line* BIST is performed simultaneously with normal functional operation of the CUT and is usually accomplished using coding techniques or duplication and comparison. When an *intermittent* or *transient* error is detected, the system will correct the error on the spot, *rollback* to its previously stored system states, and repeat the operation, or generate an interrupt signal for

repeated failures [8]. In *non-concurrent on-line* BIST, testing is carried out while a system is in an idle state. This is often completed by executing diagnostic software routines (macrocode) or diagnostic firmware routines (microcode) [11]. The process of testing can be interrupted at any time so that normal operation can resume.

Off-line BIST tests the functional circuitry in not operational mode. This approach does not detect any *real-time errors* but is widely used in the industry to perform functional testing at the system, board, or chip level so as to ensure product quality.

Functional off-line BIST tests the functionality of the circuitry based on a functional specification and usually employs functional or high-level fault model. Typically, such a test is performed using diagnostic software or firmware.

Structural off-line BIST tests the structure of the CUT. An explicit structural fault model may be used. Fault coverage is based on detecting structural faults.

2.5.1 General architecture of BIST

Basic components of a typical logic BIST scheme [11] using the *structural off-line* BIST technique are illustrated in Figure 2.1.

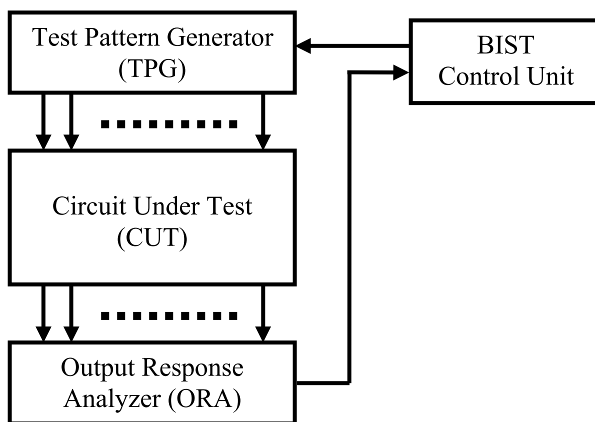


Figure 2.1 *Basic BIST scheme*

A *test pattern generator* (TPG) feeding a large number of scan chains is usually built using a *linear feedback shift register* (LFSR). Similarly, an *output response analyzer* (ORA), which generates final signatures, is also built using a LFSR. A *BIST control unit* controls testing. It initializes the TPG and the ORA, counts generated patterns and sets up end of testing signal. When the TPG is initialized it generates sequences of bits (test patterns) that are shifted into inputs of the CUT.

Then the output response of the CUT is shifted out of the scan chains and captured by the ORA. The ORA generates the signature that includes compressed output responses of the CUT during the BIST session. When output responses for all test patterns are got the final signature is compared against a fault-free signature to ascertain pass/fail for the CUT. More detailed review of methods for generating test patterns and signatures using LFSR are considered in the next section. An understanding of the TPG and the ORA concepts is essential, since because of using LFSR techniques the BIST has its benefits as well as drawbacks.

There are a number of advantages of using the *structural off-line* BIST technique rather than conventional scan [8]:

- BIST can effectively detect defects on the board or system and provide diagnostic information as required; it can run a test at any time and does not need an external tester to be present.
- Capability to perform at-speed testing, which is inherent in BIST and can be used to detect many delay faults. Applying BIST, the origin of errors can be easily traced back to the chip; some defects are detected without being modeled by software.
- Low-cost test solution since test application time, tester memory requirements and tester investment costs are reduced, because most of the tester functions reside on-chip itself.

However, there are also disadvantages associated with this approach since the CUT must comply with additional stringent *BIST-specific design rules* [11].

2.5.2 Linear Feedback Shift Register

A *linear feedback shift register* (LFSR) is a *shift register* whose input bit is a linear function of its previous state. There are two ways to implement LFSRs [11]: external feedback (Figure 2.2) and internal feedback (Figure 2.3).

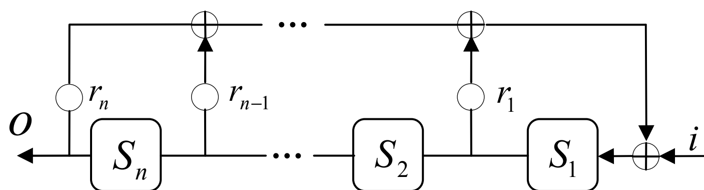


Figure 2.2 *Type 1 (external-XOR) LFSR*

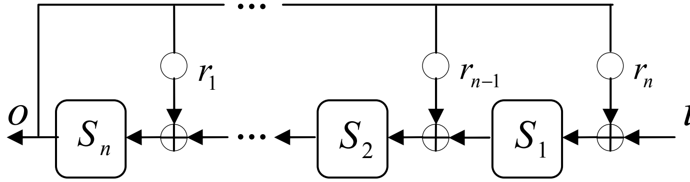


Figure 2.3 *Type 2 (internal-XOR) LFSR*

The cells in Figure 2.2 and Figure 2.3 are clocked D flip-flop triggers, where S_n is a current state of an n trigger (can be '0' or '1'). Moreover, r_n is a binary constant, and $r_n = 1$ implies that a connection exists, while $r_n = 0$ implies that no connection exists. When $r_n = 0$ the corresponding XOR (*Exclusive OR*) gate can be replaced by a direct connection from its input to its output. In order to better understand how to compute the output and the states of flip-flops, necessary formulas are presented below:

External feedback:

$$S'_1 = r_1 \cdot S_1 \oplus r_2 \cdot S_2 \oplus \ldots \oplus r_n \cdot S_n \oplus i$$
$$S'_2 = S_1$$
$$\ldots\ldots\ldots$$
$$S'_n = S_{n-1}$$
$$o = S_n$$

Internal feedback:

$$S'_1 = r_n \cdot S_n \oplus i$$
$$S'_2 = r_{n-1} \cdot S_n \oplus S_1$$
$$\ldots\ldots\ldots$$
$$S'_n = r_1 \cdot S_n \oplus S_{n-1}$$
$$o = S_n$$

where S – a current state of a register ('0' or '1'); S' – the next state of a register; i – input (note: usually input is omitted, i.e. '0'); o – output; $\oplus$ – XOR.

Analyzing these formulas is not difficult to come to a conclusion that all outputs and states of flip-flops of LFSR are determined by feedback coefficients (the values of r) and the initial state that is called the *seed* of flip-flops. The polynomial of feedback coefficients is also known as the *characteristic polynomial* (P_x) of the LFSR and is presented below:

$$P(x) = 1 + r_1 \cdot x + r_2 \cdot x^2 + \ldots + r_n \cdot x^n$$

where n – the degree of the characteristic polynomial which equals to a number of bits in a n-bit LFSR pattern. If the input of a LFSR is omitted ($i = 0$) then an all zeroes state is invalid for the LFSR as the state would never change if all bits are '0'. Therefore the maximum number of unique patterns that an n-bit LFSR can generate equals to $2^n - 1$. The characteristic polynomials of an n-bit LFSR which results in the generation of maximum possible unique patterns $2^n - 1$ are known as

primitive polynomials. The *primitive polynomials* are valid for both types of LFSRs. *The reciprocal polynomial* $P^*(x)$ of a *primitive polynomial* is also *primitive*. $P^*(x)$ is defined by the following equation:

$$P^*(x) = x^n P(1/x) \quad \text{i.e.} \quad P^*(x) = r_n + r_{n-1} \cdot x + r_{n-2} \cdot x^2 + ... + x^n$$

2.5.3 LFSRs used as pseudorandom test pattern generators

Sequences generated by LFSRs that apply primitive polynomials are called *pseudorandom sequences* [11], since they have many identical properties of random sequences. However, they are *pseudorandom*, not *random*, since they are *periodic* and *deterministic*. Some of these properties are listed below, where the sequence of maximum length $2^n - 1$ produced by n-bit LFSR using a primitive polynomial is referred to as an *n-sequence*:

- in *n-sequence* the number of 1s differs from the number of 0s by one
- *n-sequence* produces an equal number of runs of 1s and 0s
- in every *n-sequence*, one half the runs have length 1, one fourth have length 2, one eight have length 3, etc., until the fractions are integral number of runs

These properties of randomness make feasible the use of LFSRs as test sequence generators. Although, LFSRs represent the simplest and most commonly used pseudorandom TPG hardware, the efficiency of LFSR is far from optimum in terms of fault coverage and testing runtimes [23]. LFSR based test is usually up to several orders of magnitude longer than the test generated externally by a model-based ATPG. In general, *pseudo-random pattern generator* (PRPG) fault coverage has such properties like fast initial growth and too long time to complete. This is mostly caused by existence of *random-pattern resistant faults*, i.e. *hard-to-test faults* (HTTF), which can be tested only by a small number of test patterns. For example, consider a 5-input AND gate. So as to test inputs for SA0 a single pattern exists. The probability of generating the test pattern that contains all 1s will be equal to $1 / 2^5 = 1 / 32 = 0.03125$ since standard LFSR produces 1 or 0 on each output line with equal probabilities that is 1/2. Thus, there is a need to generate test patterns that have different distributions of 0s and 1s. As the result, there are many approaches proposed that target improvement of PRPG efficiency.

A lot of research was devoted to study of alternative PRPG types that have better fulfilling criteria of randomness compared to LFSR. These are, for example, *Cellular Automata* [24] and GLFSR [25]. However, the randomness has been only empirically proven to improve the quality of testing. On the contrary, larger designs, especially those that contain HTTF, need special treatment. As the result, methods based on the idea of generating weighted pseudorandom patterns that target HTTF were proposed [26]. The essence of this technique consists in adding

extra circuitry to PRPG outputs in such a way that the probability of generating 0s and 1s would be misbalanced on selected outputs to achieve fault coverage improvement.

The efficiency could be even more considerable when combining directly PRPG and ATPG patterns together, which is done, for instance, in *bit-flipping* BIST approach. Similarly to the previous technique, it uses a special circuitry that modifies specified bits of selected PRPG patterns in such a way that these patterns become equivalent to ATPG patterns that target HTTFs [27]. Another approach, called *reseeding*, allows generating several PRPG sequences where each one is optimized to detect a certain number of HTTFs [28] [29]. The final test session then consists of several subsequences, where each one starts with a special preselected *seed* loaded from memory. Some approaches replace also the LFSR *polynomial* as well as the seed [28].

The simplest way of mixed-mode BIST is represented by Hybrid BIST technique [30] where the whole test sequence is split into two parts – PRPG and ATPG. First, PRPG is set to run for a limited time, and then ATPG patterns are loaded from memory with the goal to cover remaining HTTFs. Hence, the less memory resources can be allocated, the longer sequence of pseudorandom patterns must be generated. From the hardware requirements aspect, it is low-cost solution since the patterns sources must be switched over only once. However, when memory resources are strictly limited, this method becomes impractical.

2.5.4 LFSRs used as signature analyzers

The advantage of using LFSR for signature analysis is based on the theory of polynomial division, where the "remainder" left in the register after completion of the test process corresponds to the final signature [11].

Consider the internal-XOR LFSR shown in Figure 2.4a, where the input sequence is represented by the polynomial $G(x)$ and the output sequence by $Q(x)$. The highest degree of the polynomials $G(x)$ and $Q(x)$ correspond, respectively, to the first input bit to enter the LFSR and the first output bit produced after n clock cycles, where n denotes the degree of the LFSR. If the *seed* of the LFSR contains all 0s, then the final state of the LFSR can be presented by the polynomial $R(x)$. Thus, all above-mentioned polynomials are related by the following equation:

$$\frac{G(x)}{P^*(x)} = Q(x) + \frac{R(x)}{P^*(x)}$$

where $P^*(x)$ is the *reciprocal characteristic polynomial* of the LFSR. The $P^*(x)$ is used because the input sequence corresponds to the first bit of the input stream rather than the last bit. Hence, LFSR performs *polynomial* division on the input stream by the *characteristic polynomial*, producing an output stream that corresponds to a *quotient* $Q(x)$ and a *remainder* $R(x)$.

Figure 2.4 illustrates *polynomial division*, where a single-input *signature analyzer* (Figure 2.4a) has the reciprocal polynomial $P^*(x) = 1 + x + x^3$ and the input sequence is 1110001 (the data are entered in the order shown). The input sequence and its corresponding polynomial are shown in Figure 2.4b. In Figure 2.4c a simulation of the polynomial division of the input by the LFSR is illustrated. The first three bits of the output response are ignored since they are independent of the input sequence. It is seen that $R(x) = 1 + x^2$ and $Q(x) = x^2 + x^3$.

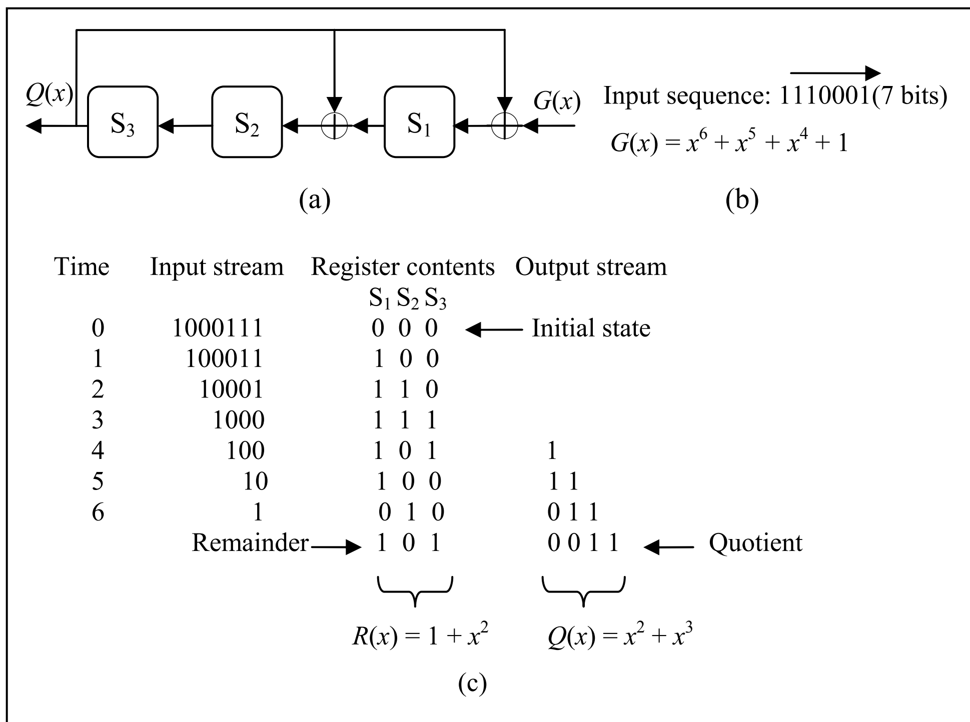


Figure 2.4 *Polynomial division*

To check correctness of the result corresponding calculations must be performed: $P^*(x) \times Q(x) = (x^3 + x + 1) \times (x^3 + x^2) = x^6+x^5+x^4+x^3+x^3+x^2 = x^6+x^5+x^4+x^2$ and $P^*(x) \times Q(x) + R(x) = (x^6+x^5+x^4+x^2) + (1+ x^2) = x^6 + x^5 + x^4 + 1 = G(x)$.

Signature analysis is the most popular method employed for test data compression because it compresses the data significantly and at the same time produces a small degree of masking [11]. If initial data is m bits and degree of a LFSR is n bits then compression coefficient is m/n; so if $m >> n$, then the compression is very efficient. At the same time, the number of data streams that produce a specific signature is $2^m / 2^n = 2^{m-n}$. For a particular fault-free response, there are potential $2^{m-n} - 1$ erroneous data streams that could produce the same

signature. Since there are $2^m - 1$ possible erroneous response streams, the masking error streams probability is the following:

$$P = \frac{2^{m-n} - 1}{2^m - 1} \cong 2^{-n}$$

where approximation is valid for $m >> n$ .Thus, the probability of no masking is $1 - 2^{-n}$, and a 16-bit signature analyzer may detect $(1 - 2^{-16})\times100 = 99.9984\%$ of the erroneous responses.

On the other hand, the final signature generated at the end of BIST cannot reveal failing patterns since it contains only *sum of remainders* of the CUT responses to a sequence of test patterns. In order to identify a failing pattern it is necessary to find two equal sequences of test patterns, one of which has one extra test pattern at the end of sequence. Additionally, the first sequence must pass a test and the second one must fail. Then, it can be concluded that the last pattern from the second sequence is the failing one. This approach requires a lot of tests to be run that increase diagnostic procedure application time significantly. Hence, the optimal search algorithm should be employed.

2.6 Logic diagnosis

Logic diagnosis is the process of narrowing down the possible locations of the defect if a logic circuit fails a test. When the candidate locations can be reduced down to possibly only a few, subsequent physical defect analysis becomes more facilitated when the search for the root causes of failure is performed. For *integrated circuit* (IC) products, logic diagnosis is crucial since it can ramp up the manufacturing yield and in some cases can reduce the product debug time as well. Typically, a successful IC product goes through two manufacturing stages [8]: (1) prototype stage and (2) high-volume manufacturing stage.

In the prototype stage, a small number of samples are produced to validate the functionality of a design on the tester and on prototype boards. During this stage, the prototype samples could fail badly due to design bugs or unstable manufacturing processes. Some of the reasons are outlined below [8]:

- *Misunderstandings about the functionality*. Since a complex product is generally defined or built by multiple engineers, there can be ambiguities, inconsistencies, and contradictions in the specifications. This can result that the actual gate-level netlist or *hardware-description language* (HDL) model may not conform to the desired specification under certain scenarios or the specification may simply have been misinterpreted.
- *Circuit marginality and timing failure issues*. Fabricated silicon may not operate properly at certain supply voltages and temperatures, or may not execute as fast as expected based on timing simulations.

- *Inappropriate layout design*. For today's nanometer technologies, the actual geometries of the devices and interconnecting wires fabricated on silicon deviate from the drawn layout due to light diffraction effects in the lithography process, since the light used has a much longer wavelength than the geometries that have to be printed.

After a design has passed the prototype stage, design bugs and circuit marginality issues are mostly resolved and the product can ramp up to high-volume production. At the early stage of manufacturing the yield could be low or fluctuating. Therefore yield improvement is essential and it can be accomplished by tuning the fabrication process. At this stage, the chip failures are mainly due to manufacturing imperfections. Some of these failures are permanent and therefore catastrophic, and some are parametric due to process variations.

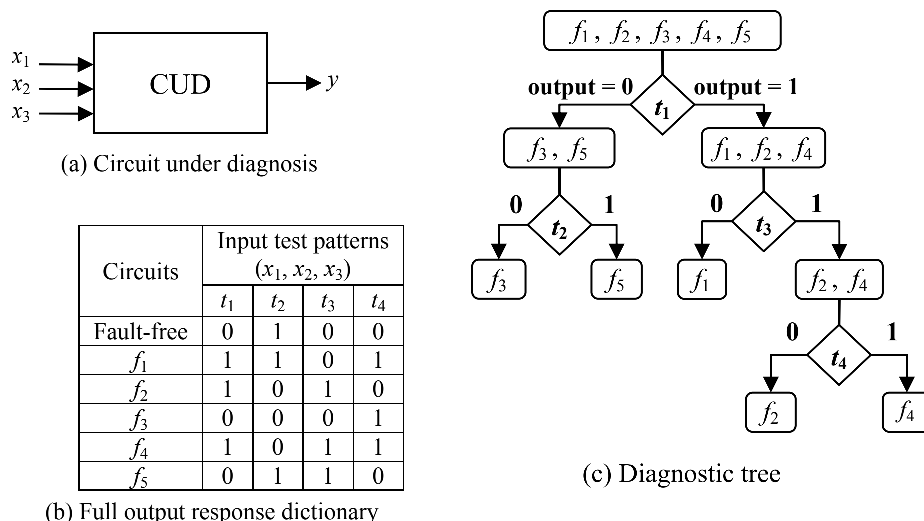
The problem of diagnosis consists in locating physical fault(s) using a structural model of a failing chip. The chip fails if its observed behavior differs from the expected behavior of a fault-free model (which is a gate-level circuit or a transistor schematic). The fault-free model is referred to as the *circuit under diagnosis* (CUD). Further, some essential definitions applied in fault diagnosis are presented [11]. *Diagnostic resolution* refers to the degree of accuracy to which faults can be located, i.e. it refers to the number of faults that are listed as fault location candidates as a result of diagnosis. If the list of candidate faults contains only *functionally equivalent* faults, then no external testing experiment can distinguish among these faults. The partition of all the possible faults into distinct sets of functionally equivalent faults defines the *maximal fault resolution* which characterizes an inherent diagnosability of the system. The *faults resolution of a test sequence* reflects its capability to distinguish among faults and is bounded by the maximal fault resolution. A test (sequence) that achieves the maximal fault resolution is said to be a *complete fault location test*. A test vector (pattern) is referred to as a *failing test vector* if it creates a mismatch at any output of CUD and a failing chip.

In logic diagnosis, the failing chip is like a black box that cannot be analyzed. The best can be done is to reason upon the CUD. Further, it is assumed that the CUD has been implemented with full-scan and its functionality is represented as a combinational gate-level circuit. In combinational logic diagnosis is assumed that the faults to be identified are located within the combinational logic. Hence, the flip-flops and the scan chains are assumed to be fault free. Thus, fault diagnosis can be approached in two different ways: *cause-effect analysis* and *effect-cause analysis* [11].

2.6.1 Cause-effect analysis

The *cause-effect* technique begins by mapping the causes of failure to a specific fault type, e.g. *stuck-at fault* (SAF) model. By intensive fault simulation *fault tables* or *fault dictionaries* are built. Once the fault dictionary is available, the

effect (or syndrome) of the failing chip is analyzed using dictionary look-up. If this look-up process is successful, the dictionary indicates the corresponding fault(s) in the CUD. Thus, cause-effect fault diagnosis is based on fault dictionaries and can be characterized as an analysis that starts with possible causes (faults) and determines their corresponding effects (responses).



Circuits	Input test patterns (x_1, x_2, x_3)			
	t_1	t_2	t_3	t_4
Fault-free	0	1	0	0
f_1	1	1	0	1
f_2	1	0	1	0
f_3	0	0	0	1
f_4	1	0	1	1
f_5	0	1	1	0

(b) Full output response dictionary

Figure 2.5 *Example of cause-effect diagnosis*

Consider a CUD that has three inputs $\{x_1, x_2, x_3\}$ and one output y, as shown in Figure 2.5a. Assume that four test vectors are generated in advance $\{t_1, t_2, t_3, t_4\}$. Based on the single stuck-at fault assumption, the fault universe will be $\{f_1, f_2, f_3, f_4, f_5\}$ after equivalent fault collapsing. Figure 2.5b shows the full response table of output signal y obtained by complete fault simulation of the fault-free circuit and the five faulty circuits. Each row corresponds to either fault-free or faulty circuit, whereas each column depicts the responses to the corresponding test vector. From the simulation results is concluded that the test set has 100% fault coverage. Further, a simple fault dictionary will be build, to assist the cause-effect diagnosis process.

A possible fault dictionary is shown in Figure 2.5c (this type of dictionary is called a *diagnostic tree*), which aids to refine the *fault candidates* iteratively [8]. The initial candidate set contains all faults. After examining the response of the failing chip to the first test vector t_1, the candidate set can be narrowed down to either $\{f_1, f_2, f_4\}$ or $\{f_3, f_5\}$ candidate sets. The refinement continues until the cardinality of each candidate set has been reduced to 1 or the responses of all test vectors have been examined. The overall diagnosis process consists in passing the path from the root of this tree to one of its leaf nodes that represent the final fault candidates. For example, if the response of a failing chip at output signal y under

the four test vectors $\{t_1, t_2, t_3, t_4\}$ is $\{1, 0, 1, 1\}$, by traversing the diagnosis tree can be immediately deduced that the only fault f_4 could be the cause of the failing circuit.

It may take a lot of time and memory to construct the fault dictionary. However, once the dictionary is built, fault analysis is usually fast. Because the fault dictionary is built only once and in the pre-diagnosis phase, the overall diagnosis process is computationally efficient. However, in practice this approach could be limited by a number of problems.

One problem is the large computational effort involved in building fault dictionaries for large circuits tested with long test sequences. To reduce computational effort, in fault simulation is employed *fault dropping*, i.e. the detected faults are dropped from the set of simulated faults (typically on first detection). Hence, all the faults detected for the first time by the same vector at the same output will produce the same signature and will be included in the same equivalence class. Thus, early fault dropping usually results in lower diagnostic resolution. Therefore, a trade-off between computation time and diagnostic resolution can be achieved by dropping faults after $n > 1$ detections. Moreover, initially a fault dictionary records every output response of each modeled fault at each clock cycle. Without proper compaction, the size of a fault dictionary is proportional to the product of three factors $F \cdot T \cdot O$, where F is the number of modeled faults, T is the number of test vectors, and O is the number of outputs, which can lead to extremely large storage requirement, so-called *dictionary size problem* [8]. The entire dictionary also has to be regenerated even if a small logic change is made. To reduce the amount of data used for fault location, a modern fault dictionary does not store the entire response caused by the fault, but only a *signature* usually consisting of list of errors contained in response. Moreover, with proper compression techniques, this problem can be relieved to some extent [31] [32] [33]. However, the excessive storage requirement and the inability to scale to ever-larger circuits still pose a serious limitation to fault dictionary based approach.

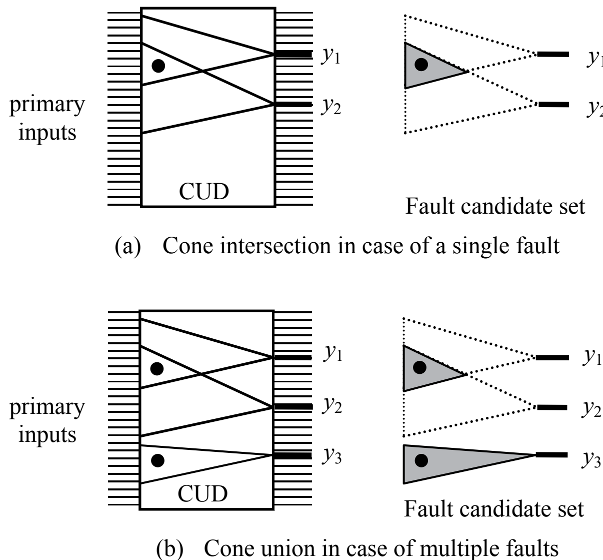
Another problem is that a fault dictionary is constructed only for a specific fault universe. If the CUD truly contains only modeled faults, then the diagnostic result is highly accurate. A fault that is not equivalent under the applied test sequence to any of the simulated faults cannot be located via the fault dictionary, because its corresponding response does not match any response computed by fault simulation. Moreover, realistic defects may not behave as modeled faults and can easily lead to misleading results.

2.6.2 Effect-cause analysis

Unlike the fault-dictionary-based paradigm, *effect-cause analysis* directly examines the responses (effects) of the failing chip to derive the fault candidates (causes) through *Boolean* reasoning on the CUD [8].

Structural pruning techniques are often used as the first-step process in effect-cause analysis that can narrow down the potential *fault candidate area* in the CUD [8]. Firstly, the approach identifies *mismatched outputs* in the CUD for given sequence of test vectors. The primary output in the CUD is called a *mismatched output* whereas the corresponding primary output in the failing chip is called a *failing output*, if there exists a test vector *t* which when applied to both the CUD and the failing chip produces different binary values at corresponding outputs. Secondly, the *fanin cones* of *mismatched outputs* are estimated that define the *fault candidate area* in the CUD. The *fanin cone* of an output in the CUD refers to the collection of the logic gates that can reach this output structurally.

Depending on the number of faults in the failing chip, there can be employed *cone intersection* or *cone union* to prune out those logic gates that could not possibly produce the faulty behavior [34], as illustrated in Figure 2.6. If there is only one fault, then the *intersection* of the fanin cones of the mismatched outputs is taken and the resulting area of gates is the fault candidate area (Figure 2.6a). On the other hand, if there is more than one fault in the failing chip, then *cone union* should be accomplished instead (Figure 2.6b). The reason is that every gate in the fanin cone of any mismatched output could now be responsible for the observed error partially, if not completely. The pruning capability of *cone intersection* is much more effective than that of *cone union*. However, the number of faults, that force the chip to fail, is unknown in advance before the diagnosis process. So, *cone union* is a conservative and safer technique, whereas *cone intersection* could lead to an empty fault candidate area if there are multiple faults.



(a) Cone intersection in case of a single fault

(b) Cone union in case of multiple faults

Figure 2.6 *Structural pruning techniques*

When structural pruning is completed and fault candidate area determined, a more accurate procedure to pinpoint the fault locations is applied. *Backtrace* is one such procedure which corresponds to *functional pruning* technique. *Functional pruning* disqualifies candidates from the fault candidate area by examining the signal values inside the CUD with simulation [8]. *Backtrace* is similar to *critical path tracing*, which was originally proposed by [35] for fast fault simulation and then subsequently applied to logic diagnosis [36]. The *backtrace algorithm* iterates through each failing test vector performing, first, a fault-free simulation on the CUD and then checking the mismatched outputs one at a time. From each mismatched output, it traces the CUD backward to identify the signals that can cause the output mismatch.

Figure 2.7 demonstrates the *backtrace algorithm* on a simple circuit. Here is made no distinction between a logic gate and its output signal. The figure shows a trace starting from mismatched output signal *e* where the fault candidate area is marked by bold lines. At the end of the *backtrace* algorithm there will be one fault candidate set for each mismatched output under a specific failing test vector. There will be an issue in combining these fault candidate sets into a final set. In case of a single fault assumption, the intersection of all of these fault candidate sets will indicate the final fault candidate set. If the final set occurs to be empty, then multiple faults should be implied to present in the failing chip.

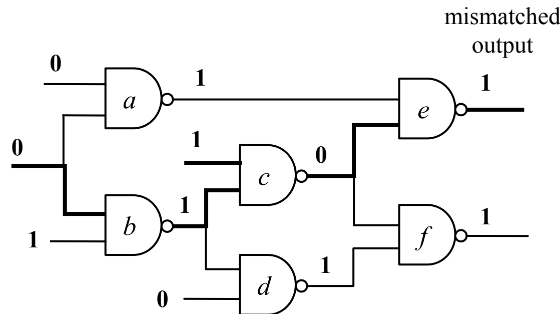


Figure 2.7 *The backtrace algorithm demonstration*

The *backtrace algorithm* is generally efficient, but in some cases it may not be accurate enough. To address this issue, [37] pioneered an alternative method, referred to as the *inject-and-evaluate paradigm*. Computationally, it performs a sequence of fault injections and evaluations to diagnose a failing circuit.

2.7 State-of-the-art

2.7.1 BIST environment challenges for fault diagnosis

As process technologies shrink and designs become more complex, *built-in self-test* (BIST) is gaining increasing acceptance as an industry-wide test solution [14], because it provides a low-cost solution to both test generation and test application [38], offers the promise of low hardware overhead with the clear advantage of at-speed testing (delay testing) [39], enables testing at the operation frequency of *circuits under test* (CUT) and reduces test application time in comparison with *automatic test equipment* (ATE).

Despite such benefits, the BIST approach has not been adopted as the primary test methodology, because of weak diagnostic capabilities [38]. A problem with it is that the signature provided by *output response analyzer* (ORA) at the end of the test session does not contain enough diagnostic information, either to identify failing vectors or to precisely identify error-capturing scan cells. The pass/fail information obtained from ORA is usually insufficient to diagnose the failure via effect-cause analysis [40]. Thus debug in BIST environment is complicated.

To ensure its overall success, a BIST environment must be able to provide similar diagnostic capability as a conventional scan-based external testing environment. Any new method proposed to eliminate this drawback of BIST technique is faced with following challenges in the diagnosis process to be solved [39]:

- achieving full diagnostic information, i.e. detecting all faulty items (whether scan cells, test vectors or logic blocks)
- minimizing diagnostic time, since this translates to reduction in total testing time
- minimizing hardware overhead, i.e. the amount of hardware needed to support *built-in self-diagnosis* (BISD)

Early works on BISD have focused on extracting diagnostic information hidden in the BIST signature based on identification of fault-detecting test vectors [41] [42] [43]. However, if there are faults that are detected by a number of vectors, aliasing problems make it impossible to place an accurate diagnosis [42]. Therefore, the researchers have attempted to collect more information by repeating the same test [*doubling*, *jumping*], by adjusting *signature analyzer* (SA) parameters or observation outputs or by increasing signature register size [44] [45] [46] [47]. However, the previous attempts based on the analysis of a single signature have failed to provide effective methods for current designs [48]. On the other hand, SA partitioning based diagnosis schemes [45] [46] [47] [49] have been introduced that have proven highly effective for large industrial circuits.

Research on identification of fault-embedding scan cells in scan-based BIST design has concentrated on increasing the diagnostic information through multiple applications of the same test, each time modifying the way that the test responses are compacted. Observation of the test responses can be modified by either changing the outputs to be observed through partitioning schemes [48] [49], or changing the signature compactor [47]. The approaches based on multiple repeating of full test sequences lead to long diagnosis procedures, and the control circuits needed for repeated partitioning or repeated changing of signature analyzers lead to higher area overhead.

Another approach is to organize the diagnostic procedure as a sequence of selected test sequences with the goal to identify failed test patterns. These approaches are using cause-effect strategy and are implemented as a classical search procedure which can be optimized to reduce the total diagnosis time. The drawback of this type of known approaches is the high amount of memory space needed to store the pre-computed fault tables.

As logic BIST is increasingly being adopted to improve test quality and reduce test costs for rapidly growing designs, the search for efficient methods for BISD that provide the same benefits as BIST is relevant. Although many methods have been offered [38] [39] [40] [6], there is still a potential to improve them. Thus the problem of looking for efficient fault diagnosis methods remains.

2.7.2 The strategies for diagnosis

Diagnostic analysis is based on two main principles: cause-effect or effect-cause approaches [8]. *Cause-effect analysis* [31] is based on pre-computed fault dictionaries whereas *effect-cause analysis* [50] [51] is based on processing the test responses of a circuit (effect) in order to locate the fault (cause).

Cause-effect analysis is also called a fault-dictionary based paradigm or combinational diagnosis approach [52]. As designs grow in complexity, dictionary-based diagnosis becomes infeasible due to prohibitively large dictionary sizes. The disadvantage of the cause-effect approach lies also in the need of fault models or fault lists, and in the inability to handle multiple faults.

Effect-cause analysis, while not using expensive fault dictionaries, requires, however, significantly higher computational power during diagnosis, compared to the cause-effect approach. For every failing circuit, an expensive diagnostic simulation of the circuit has to be processed. The test responses have to be analyzed by using fault simulation and backtracking to identify the causes [50] [53] [54] [55]. Because of sequential character of reasoning this approach is called also *sequential fault diagnosis*. Effect-cause analysis is superior to the fault-dictionary-based paradigm at least in the following aspects [8]: a) it does not assume a fault model and thus is more suitable for handling non-stuck-at-faults; b) it can be easily adapted to cases where there are multiple faults in the failing chip.

Effect-cause algorithms are simple when the single fault assumption is adopted. In this case intersections of input cones of failing outputs are calculated [56], or *backtrace critical paths* from failing outputs are processed [57].

Computational and storage requirements for fault dictionaries are high due to the large number of faults, outputs and test patterns. Reduction in size is possible by storing only subsets of the dictionary, at the expense of reduced diagnostic resolution [58] [33]. Also, a combination of the two approaches, *cause-effect* and *effect-cause* analysis, can have a synergy to overcome the pre-diagnosis and post-diagnosis simulation costs.

2.7.3 Fault models

Traditional approaches to the cause-effect fault diagnosis lay on the *stuck-at-fault* (SAF) model. Due to the advances in manufacturing technologies, more and more defects lead to failures that can no longer be modeled by classical stuck-at faults. Numerous actual failures exhibit timing or parametric behaviors that are not represented by stuck-at faults. Therefore, many researchers have focused on developing new fault models for particular types of failure mechanisms like *signal line bridges* [59] [60], *transistor stuck-opens* [61] [62] or failures due to changes in circuit *delays* [63].

However, the variety of possible physical defects or error causes in an electronic circuit is practically infinite, and therefore all the possible faults are not countable. On the other hand, the know-how about realistic physical defects in microelectronic circuits is quickly getting obsolete. New semiconductor processes introduce new types of defects and fault effects. New failure mechanisms are emerging that are not fully understood. This makes defect model based fault diagnosis extremely difficult, and the traditional diagnosis methods based on using different error or fault models are becoming outdated. To overcome the difficulty of counting the physical defects in microelectronic circuits and bugs in designs, new fault model free approaches to debug and diagnosis are emerging.

Consequently, another trend has been to develop general fault modeling mechanisms and corresponding test tools that can effectively analyze arbitrary fault types like in [64] where *D-cubes* are used to model any arbitrary change in the logic function of a circuit block. A generalization of this approach has been found in the *input pattern fault model* [65] and in the *pattern fault model* [66] which can model any arbitrary change in the logic function of a circuit block, where a block is defined to be any combinational subcircuit described at any level of the design hierarchy.

A similar pattern related fault modeling approach called *functional fault model* was proposed earlier in [67] for the module-level fault diagnosis in combinational circuits based on solving systems of Boolean differential equations. The functional or pattern fault model allows an arbitrary set of signal lines to be grouped into

activation conditions for a single fault site, allowing a variety of fault types to be modeled. The functional faults can be either static or dynamic [68].

In [69], a similar model called *conditional fault* was proposed for test generation purposes, and in [70] for diagnosis purposes. A conditional fault allows additional signal line objectives to be combined with the detection requirements of a particular fault or physical defect.

The *single location at a time* (SLAT) technique [71] [72] [73] relaxes the single fault assumption, and is a fault model independent approach which uses the SAF model only to localize the suspected area of the circuit. The drawback of the SLAT paradigm based on the conditional SAF model is the fact that only patterns with SLAT property are used, all the other patterns are not taken into account. An adaptive diagnosis approach as an extension of the SLAT technique is proposed in [74] [75]. It combines a novel effect-cause pattern analysis with high-resolution ATPG.

2.7.4 Fault simulation

Fault models are used in test generation and fault diagnosis whereas the efficiency of both relies heavily on the efficiency of the fault simulation. For some of the advanced fault models dedicated fault simulation methods has been developed, e.g. symbolic X-fault simulation [76], simulation of resistive bridging faults based on resistance intervals [77] etc.

For most of the proposed fault models it is possible to divide the fault simulation into two phases:

- traditional SAF simulation to determine which nodes in the circuit are “active”, i.e. observable via fault propagation to primary outputs at the given test pattern
- dedicated defect analysis according to the selected advanced fault model to determine which realistic defects can influence the signals at observable (active) nodes

The final efficiency of the fault simulation is highly depending on the speed of the first phase – SAF simulation.

2.7.5 A hierarchical approach to fault diagnosis

Rapid advances in the areas of nanoscale electron technology and design automation tools enabled engineers to design larger and more complex integrated circuits. On the other hand, the increasing integration densities pose severe problems with respect to the quality assurance. The quality and reliability of microelectronic circuits depend essentially from the efficiency of debug and diagnosis of failures in circuits. Traditional approaches to automate the diagnostic

processes focus on gate-level designs [78] [79] [80] [74] [57]. However, due to the continuous increasing of the gate count in circuits under diagnosis, the gate-level methods are becoming less efficient. This phenomenon, on one hand, and the fact that the conversion of a *register-transfer level* (RTL) design into gate level is fully automated, on the other, results in the efficiency of gate-level based methods [81]. Also, as most errors are designer-introduced [82], the ability of directly pinpoint the errors in *hardware-description language* (HDL) designs can provide the design engineer with a better understanding of the nature of the design failure.

Two main trends can be observed when searching solutions for the problems of testing and diagnosis: defect-orientation, and high-level modeling [8]. The trend towards high-level modeling helps to cope with the complexity, but moves even more away from the real life of physical defects and, hence, from accuracy of diagnosis. To handle adequately defects in nanoscale technologies, new fault models and defect-oriented diagnosis methods should be used. But, the defect-orientation is increasing again the complexity. To get out from the deadlock, these two opposite trends – high-level modeling and defect-orientation – should be combined into hierarchical approaches.

2.7.6 E-learning in digital fault diagnosis

The importance of testing and fault diagnosis in technical systems as a teaching objective is often underestimated in engineering education. The more complex electronic systems are getting the more important it becomes to solve the problems of testing and fault diagnosis because of the complexity of these problems and high cost of solutions. Today, design and test are no longer separate issues. Entering into the system-on-chip and network-on-chip era means that test and diagnostics must become an integral part of the electronic system design courses. However, fault diagnosis is not only an electronic systems related issue, it has an important didactic role for engineering education in general: (1) it is a method to learn how to ask right questions, (2) it develops abilities to analyze cause-effect or effect-cause relationships, (3) it forces to look for answers to the questions like what is the reason of what has happened.

The electronics world because of its inherent logic complexity could be the best objective for learning the concepts of diagnostic reasoning for any technical systems in general. It is not only a system design or manufacturing issue, it is a problem to be solved every day in the field when a system stops working because of a fault.

A student is not coming to university to be taught, he is coming to learn how the professor thinks. Students should not be asked to press simply on buttons in laboratories to get results which only confirm what they have heard in classes or what they know already. The real targets of education are: creativity, critical thinking, and problem solving skills. Therefore, teaching and learning at a university should be research oriented.

2.7.7 Conclusions

- Despite the benefits of the BIST approach as test methodology, it has weak diagnostic capabilities. Methods are needed which allow to develop embedded fault diagnosis methods on the basis of standard BIST solution, and which would provide cost-effective, fast diagnosis procedures and high diagnostic resolution.
- Broadly spread fault dictionary based cause-effect diagnosis is a fast and efficient approach; however it is not scaling, and cannot be used for complex digital systems. High-level or hierarchical methods are needed to cope with the growing complexities of fault dictionaries to make the cause-effect approach scalable.
- The drawback of effect-cause analysis is significantly higher computational power needed for diagnosis, compared to the cause-effect approach. A combination of both, cause-effect and effect-cause paradigms coupled with hierarchical approaches could be a way to develop cost-effective and fast diagnosis procedures and high diagnostic resolution.
- Two main trends in testing and diagnosis – defect-orientation, and high-level modeling – are conflicting with each other. High-level modeling helps to cope with the complexity inherent in physical defect level modeling, but provides poor accuracy of diagnosis. To cope with the difficulties of handling defects in nanoscale technologies fault model free diagnosis should be taken as a new paradigm of fault diagnosis in today's digital systems. Two opposite and conflicting trends - high-level modeling and defect-orientation - should be combined into hierarchical approaches.
- The importance of fault diagnosis in technical systems as a teaching objective is often underestimated in engineering education. Entering into the system-on-chip and network-on-chip era means that test and diagnosis must become an integral part of electronic system design courses. There is a need for low-cost and simple tools for hands-on training in system diagnosis, to support these courses and to foster in students creativity, critical thinking, and problem solving skills.

Chapter 3

FAULT DIAGNOSIS IN THE BIST ENVIRONMENT

This chapter presents an optimized fault diagnosing procedure applicable in *built-in self-test* (BIST) environments. Instead of the known approach based on a simple bisection of patterns in pseudorandom test sequences, a novel bisection procedure is proposed where the diagnostic weight of test patterns is taken into account. Another novelty is the sequential nature of the procedure which allows pruning the search space. Opposite to the classical approach which targets all failing patterns, in the proposed method not all failing patterns are necessarily needed to be fixed for diagnosis. This allows to make a trade-off between the speed of diagnosis and diagnostic resolution. The proposed method is compared with three known fault diagnosis methods: *classical Binary Search*, *Doubling* and *Jumping*. Experimental results demonstrate the feasibility and efficiency of the approach.

3.1 Search procedures for diagnosis

In the following an overview of diagnostic algorithms in the connection of using in BIST is given. Consider the BIST environment consisting of pseudorandom *test pattern generator* (TPG) and *multiple input signature register* (MISR) as an *output response analyzer* (ORA) for *circuit under test* (CUT) as depicted in Figure 2.1.

Denote by N the length of the pseudorandom test sequence T generated by TPG, by F the set of possible faults in the CUT, by $F(t) \subset F$ the set of faults detected by

the test pattern $t \in T$, and by $T(f) \subseteq T$ the set of test patterns failed because of the fault $f \in F$. Denote by a test session (query) the procedure where a part of test sequence T is applied with the subsequent comparison of the signature in MISR with the expected one.

The diagnosis problem can be formulated as follows: Given a set F of faults, identify the subset of faults $F^* \subset F$, where in general case, the number of faults to be localized $d = |F^*|$ is unknown, using the minimum number of queries. The number of queries is directly proportional to the amount of time needed to diagnose the BIST system.

Below four algorithms are discussed: *Binary Search*, *Digging*, *Doubling* and *Jumping* that attempt to solve the diagnosis problem.

3.1.1 Binary Search

The classical *Binary Search* algorithm is based on bisection of patterns, and a large variation of this approach has been published [83] [84] [85] [86]. Consider here the case $d = 1$ where the circuit contains a single fault $f^* \in F$. The following procedure describes how to find all the failing patterns $T(f^*) \subseteq T$ within the pseudorandom sequence $[1, N]$ of test vectors T generated by BIST, if after N patterns the signature is corrupted [87]:

Algorithm 3.1: (see Figure 3.1)

1. Perform BIST for all patterns within $[1, N/2]$
2. *IF* the signature after $N/2$ patterns is correct:
   Find all the failing patterns within $[N/2+1, N]$
   ELSE
   Find all the failing patterns within $[1, N/2]$
   Load the correct seeds for the pattern $N/2+1$ into the TPG and MISR
   Find all the failing patterns within $[N/2+1, N]$

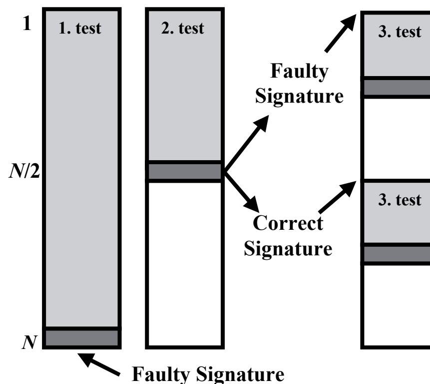


Figure 3.1 *Binary search algorithm description*

After the set of all failing test patterns $T(f^*)$ is determined the set of faults can be calculated as follows:

$$F(f^*) = \bigcap_{t \in T(f^*)} F(t) - \bigcup_{t \in T - T(f^*)} F(t)$$

containing the suspected faults which cannot be distinguished from f^*.

3.1.2 Digging

The *Digging* algorithm can be considered as an improvement to the *Binary Search*. *Digging* reduces the number of queries, especially for low values of d (number of faults in the CUT) [88] [89]. Observe that if there are two sets of suspected faults F_1 and F_2, with $F_1 \subset F_2$, then the result of the query F_1 renders the result of the query on F_2 useless. Hence, with *Binary Search* there is a potential for many queries to produce no additional information for the diagnosis process. This suggests that once a suspected set of faults $F(f^*)$ is found, the searchable fault f^* should be identified from this particular set $F(f^*)$. This process is referred to as *Digging* [89].

In general case with $d > 1$, once a fault f^* is identified, f^* is removed from F^*, and digging is resumed on the remaining items. *Digging* requires $d \times \log_2 n$ queries.

3.1.3 Doubling and Jumping

Given that the value of d is unknown, the *Doubling* algorithm attempts to estimate the value of d. If d is small then the algorithm finds large fault-free sets; otherwise, the algorithm finds small suspected fault sets. To deliver this functionality, the algorithm tests disjoint sets sizes 1, 2, 4, …, 2^i, where $i = 0, 1, \ldots$, until a suspected fault set is found. At this point, the algorithm has used $2^i - 1$ test patterns with positive results and has identified a fault by using a set of test patterns of size 2^i, using $i + 1$ queries. The algorithm then identifies the fault within a sequence of 2^i test patterns using binary search, which requires i queries.

Consequently, in a general case, the algorithm uses $2i + 1$ queries and detects 2^i items ($2^i - 1$ fault free and 1 faulty). This *Doubling* algorithm is presented in [39] [90].

An interesting modification to *Doubling* is *Jumping* [91]. Here the test sets of sizes 1+2, 4+8, ..., $2^i + 2^{i+1}$ are used until a faulty set is found. Using these "jumps" in the ordered test sequence, the algorithm identifies fault-free subsequences with $i/2$ tests instead of i tests. However, a faulty test subsequence is of size 3×2^i, rather than of size 2^i as in *Doubling*; it therefore requires more than one query on a subset of size 2^i to reduce the faulty set to either 2^i or to size 2^{i+1} with 2^i fault-free items. More detailed analysis of *Jumping* algorithm is presented in [39] [91].

3.1.4 Summary

In this section four most effective algorithms for finding failing patterns in a sequence of test patterns have been discussed. The effectiveness of the presented methods mainly depends on the number of failing patterns and their location in the sequence of test patterns. Thus in some cases one algorithm is better than other and conversely.

The algorithms of *Binary Search*, *Doubling* and *Jumping* to be used in the BIST environment were implemented to make a comparison with the proposed searching algorithm described in the next section. Since in the thesis was considered only the case d = 1 (assumption of a single fault), the *Digging* algorithm was not implemented. In fact, indirectly the main idea of the *Digging* algorithm about successively concentrating the search on current sets of suspected faults is covered by the proposed algorithm.

3.2 Diagnosis with bisection by fault coverage

3.2.1 Built-in self-diagnosis environment

In Figure 3.2 a BIST based architecture is presented which is used to conduct the embedded fault diagnosis. The environment consists of pseudorandom TPG, ORA for fixing output responses, memory for storing diagnostic data, and BIST control unit.

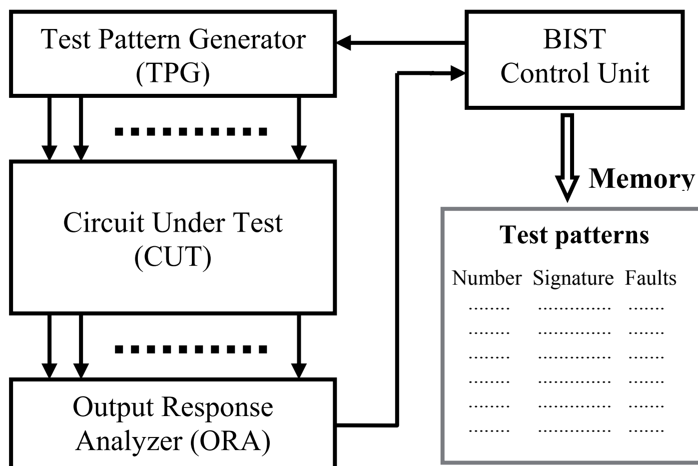


Figure 3.2 *BIST for fault diagnosis*

The proposed method is based on sequential bisection of the pseudorandom test sequence controlled by diagnostic data in the BIST memory. In other words, the

idea of bisection of faults instead of bisection patterns as used in former methods is employed. Selected patterns in the test sequence are serving as *diagnostic points* (DPs). The number of DPs is determined by trade-off between the cost of memory and diagnostic resolution. Generally, patterns which detect new faults not yet tested by previous patterns are selected as DPs.

Diagnostic data (DPs) in the memory (see Figure 3.2) consist of:

- the numbers j of selected test patterns t_j
- signatures $s(t_j)$ corresponding to the content of ORA if the test pattern t_j would be the final pattern of the test session
- the sets of faults $F(t_j)$ detected by the test patterns t_j

For each test pattern t_j can be calculated the set of faults $F(T_j)$ detected by the test sequence T_j with final pattern t_j as:

$$F(T_j) = \bigcup_{t_j \in T_j} F(t_j)$$

and the fault coverage FC_j reached by the test sequence T_j with final pattern t_j as:

$$FC_j = |\, F(T_j) \,| \,/\, |\, F \,|$$

where F is the set of all possible faults detected by the test sequence.

Besides using static tables of the numbers, the signatures and the sets of faults of test patterns, there is also a need to keep in memory a dynamic table of set of faults $F(T_j)$, a list of suspected faults and numbers of the first and the last patterns of current test sequence, to be able to run the diagnostic procedure. Moreover, extra hardware is needed to perform basic logical operations to support the proposed method (necessary logical operations will be discussed further).

3.2.2 Algorithm description

In *Algorithm 3.2*, *step – 0...100%* defines the interval of patterns used for the diagnostic test session (query) where **%** means percentage of fault coverage, i.e. *step = 100%* (F_s) means that a test sequence $T = [t_{start}\,,\, t_{step}]$ must test all suspected faults F_s . Initially, all faults in F are being suspected and therefore *step* = N (index of the last test pattern in initial test sequence). Notations *start* and *end* represent the index of a pattern in the test sequence.

Firstly, a test containing all the test patterns is applied to the CUT. In case, the test fails, i.e. the signature $s(t_{end})$ does not correspond to the expected one, a subsequence of the initial test is selected in such way that the new test sequence $T = [t_{start}\,,\, t_{end}]$ detect t_{end} = *step*/2 = *50%* of faults in F_s . The calculation of the index t_{end} is completed employing information from total fault table $\{F(T_j)\}$. If the new test detecting 50% of faults in F_s passes, i.e. the signature $s(t_{end})$ corresponds to the expected one, the list of suspected faults is updated $F_s = F_s - F(T\,[t_{start}\,,\, t_{end}])$.

The bisection procedure of searching the failing pattern t_f is continued until a failing test sequence that contains only a single test pattern ($t_{start} = t_{end}$) is found. Then the list of suspected faults $F_s = F_s \cap F(t_f)$ as well as fault tables $\{F(t_j)\}$ and $\{F(T_j)\}$ are updated. Further, a decision is made either to continue the searching procedure or to finish the diagnosis if applicable diagnostic resolution is achieved.

Algorithm 3.2:

Initial states:

1. Initial fault table $\{F(t_j)\}$ and total fault table $\{F(T_j)\}$
2. Suspected faults F_s = all faults in F
3. T = [t_{start} , t_{end}] initial test sequence, where t_{start} = 1 and $t_{end} = N$
4. *step* = 100%(F_s)

Do

5. Load correct seeds for t_{start} into the TPG and the ORA
6. Perform BIST for all patterns within T = [t_{start} , t_{end}]
   // (1) diagnose, update F_s
7. ***If*** (s(t_{end}) is correct) *// test passed*
8. $\quad F_s = F_s - F(T[t_{start}, t_{end}])$
9. ***Else*** *// test failed*
10. $\quad$ ***If*** (*start* = *end*) *// single t_f is found*
11. $\qquad F_s = F_s \cap F(t_f)$
12. ***If*** (F_s updated)
13. $\quad$ *start* = *end* + 1
    // (2) update $\{F(t_j)\}$ and $\{F(T_j)\}$, where $j \in [start, N]$
14. $\quad \{F(t_j)\}[j]$ = initial $\{F(t_j)\}[j] \cap F_s$
15. $\quad \{F(T_j)\}[j] = \{F(t_j)\}[j] \cup \{F(T_j)\}[j\text{-}1]$
16. $\quad$ ***If*** *(t_f is found)*
17. $\qquad$ *step* = 100%(F_s)
18. $\quad$ ***Else*** *// test passed*
19. $\qquad$ *step* = *step*/2 *// (3) calculate step*
20. ***Else*** *// test failed, but the failing pattern cannot be extracted yet*
21. $\quad$ *start* = *start*
22. $\quad$ *step* = *step*/2 *// (3) calculate step*
23. *end* = *step*

While ($t_{start} \leq N$)

By *Algorithm 3.2* a *diagnostic tree* (DT) can be created where the nodes represent test sessions (Figure 3.4 and Figure 3.5). Each path in DT represents a diagnostic procedure as a sequence of test sessions.

For fault diagnosis, in fact, the explicit full diagnostic tree is not needed. For the fault location, only a single path of such a tree should be created and carried out according to *Algorithm 3.2*. The last node of the path corresponds to a failing

pattern t_j which allows to determine the set of suspected faults as the result of fault diagnosis:

$$D = F(t_j) - F(T_j - t_j)$$

If the diagnostic resolution $| D |$ is acceptable the procedure can be finished, otherwise the fault diagnosis will be continued. Now it will take into account the knowledge of D in deciding the t_{start} and t_{end} patterns for further test sessions. In this procedure the same bisection algorithm will be used where the fault coverage of the patterns involved in test sessions will be updated by using D.

The search for new failing patterns to improve the current diagnosis D (to reduce the number of suspected faults in D) proceeds until either all failing patterns are found or acceptable diagnosis resolution is achieved. The maximum number of iterations equals to the number of all failing patters in the given sequence of test patterns. The main difference of the proposed method compared to [87] is in searching and processing only a part of all failing patterns to reach still acceptable resolution.

3.2.3 Example of bisection detected faults

In the following example, the problem of diagnosis will be described as a set of possible diagnostic procedures in a form of diagnostic tree. If the full diagnostic tree is given then the average length of the diagnostic procedure (the number of test sessions or queries) can be calculated. On this small example is also demonstrated how the average length of the diagnostic procedure can be reduced by the proposed method of bisection faults compared to the former method of bisection test patterns.

Consider as an example the small circuit c17 from the ISCAS'85 benchmark family [92] [93] depicted in Figure 3.3.

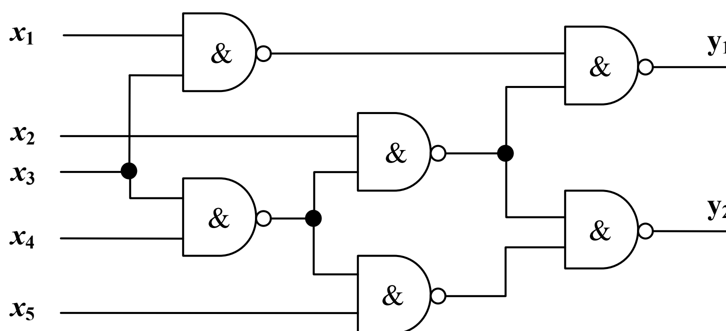


Figure 3.3 *A schematic of circuit c17*

In Table 3.1 the diagnostic data for circuit c17 generated during *stuck-at fault* (SAF) simulation of the pseudorandom test sequence are presented, where $|F(t_j)|$ denotes number of stuck-at faults tested by the test pattern t_j , $|F(T_j)| - |F(T_{j-1})|$ – number of new faults, not jet tested by previous patterns, $|F(T_j)|$ – cumulative number of tested faults, and FC_j % – cumulative fault coverage in percentage. In this example all the test patterns are considered as DPs since each test pattern detects new faults.

Table 3.1 *Fault simulation data for circuit c17*

#	Input Patterns	$\lvert F(t_j)\rvert$ # faults	$\lvert F(T_j)\rvert - \lvert F(T_{j-1})\rvert$ # new faults	$\lvert F(T_j)\rvert$ # cumulative faults	FC_j %
1	11001	5	5	5	16.7
2	10010	10	10	15	50.0
3	00100	9	1	16	53.3
4	01000	6	1	17	56.7
5	10001	10	3	20	66.7
6	00011	10	1	21	70.0
7	00111	11	4	25	83.3
8	01111	12	1	26	86.7
9	11110	11	3	29	96.7
10	11100	6	1	30	100.0

3.2.3.1 Primary diagnostic tree

Based on information about new faults detected by test patterns and cumulative fault coverage FC_j% given in Table 3.1, the proposed algorithm builds up the primary diagnostic tree to perform diagnostic procedure (fault diagnosis). According to the Table 3.1, the sequence of two first patterns testing 50% of faults should be carried out as the first test. The collected signature is compared to the expected one. If the test fails then a fault has been detected and the second test session will now consists of only the first pattern. In the opposite case, if the test passes, the new test session must test 50% / 2 = 25% of not yet tested faults. The new test will contain patterns from 3 to 6 since cumulative fault coverage of pattern 6: FC_6 = 70% − 50% = 20% is closer to 25% of faults than of pattern 7: FC_7 = 83.3% − 50% = 33.3% . The diagnostic procedure is continued until a failing test containing a single pattern is found.

In Figure 3.4 the diagnostic trees for comparing the classical binary search with bisection of patterns (Figure 3.4a) and the proposed algorithm with bisection of detected faults (Figure 3.4b) are presented. From each node in the trees the algorithm proceeds to the left if a fault is detected by the corresponding test session, and to the right in the opposite case. The numbers at the outputs of the leaves on trees correspond to the diagnostic resolution achieved by this particular procedure. The trees allow calculating the length of the diagnostic procedure.

For example, in Figure 3.4b the path to faults detected by pattern 3 lies through nodes 2, 6, 5, 4, 3 and corresponds to 5 tests with the length of 2 + (6−2) + (5−2) + + (4−2) + (3−2) = 12 clocks (test patterns). From nodes 6, 5, 4, 3 is subtracted 2 since the first test containing two first patterns (node 2) passes and therefore the following test sessions starts from pattern 3, not from pattern 1. Similarly, in classical binary search tree (Figure 3.4a) the path to node 3 lies through nodes 5, 2, 3 that corresponds to 3 test sessions and 5 + 2 + (3−2) = 8 clocks.

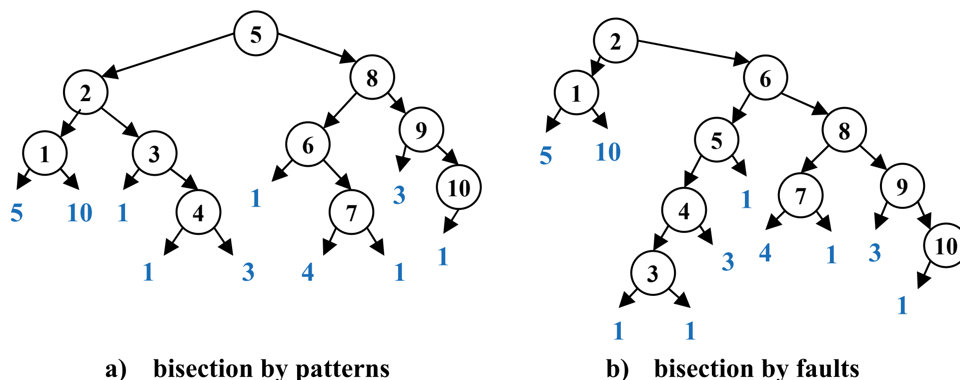


Figure 3.4 *Primary diagnostic trees for c17*

For this particular case, the classical binary search approach outperforms the proposed method. However, if to carry out deeper analysis of the diagnostic tree, it is evident that the tree based on bisection of faults is not height-balanced (shifted to the right). Considering the numbers at the outputs of the leaves on the tree that correspond to the diagnostic resolution, the path through nodes 2 and 1 detects 50% of all faults ((5 + 10) / 30 × 100% = 50%) only by 2 tests. Whereas, the *classical binary search* tree uses 3 tests – a path through nodes 5, 2 and 1. As a result, the average lengths of test sessions and clocks, that are needed to identify a failing pattern over all stuck-at faults, for the proposed method are 3.06 and 6.43, whereas for the classical binary search are 3.33 and 8.67, respectively.

For complex circuits, where a lot of test patterns needed for test and diagnosis since a large number of stuck-at faults are possible in CUT, the values of $|F(T_j)| - |F(T_{j-1})|$ will progressively decrease with growing j. This is the reason why for more complex circuits the algorithm using bisection by fault coverage will produce the diagnostic trees even more shifted to the right. This means that 50% of faults can be always analyzed by a small number of test sessions. The difference is significant when for fault diagnosis is applied a sequence of *pseudorandom* test patterns.

Because of the *right-shifted* primary diagnostic tree, the proposed method on average finds out the first failing pattern more effectively than the classical binary search tree. Although in the given example the difference between primary trees is

not evident, i.e. the average lengths of test sessions and clocks differ slightly; the superiority of the proposed method becomes obvious when examining secondary diagnostic trees, a case when the first failed pattern is already found.

3.2.3.2 Secondary diagnostic tree

Consider the case when pattern 2 fails. From the Table 3.1, the number of faults to be suspected as a result of fault diagnosis is equal to 10 (new faults detected by pattern 2), since from the number of cumulative faults $|F(T_2)|$ is subtracted faults $|F(T_1)|$ detected by pattern 1. The diagnostic resolution is weak since $10 / 30 \times 100\% \approx 33.33\%$ of all possible stuck-at faults are suspected to present. The fault diagnosis can be continued to improve the resolution. A secondary diagnostic tree should be built up to perform a search of other failing pattern. So as to support the generation of the secondary tree, initial fault table $\{F(t_j)\}$ and total fault table $\{F(T_j)\}$ must be updated according to the proposed algorithm.

Table 3.2 *Updated fault table for circuit c17*

																$F(t_j)$	$F(T_j)$	$F(T_j) - F(T_{j-1})$	FC_j %
1	X	X	X	X	X	0	0	X	X	1	X	X	X	0	0				
2	1	1	X	X	X	X	1	1	X	0	0	0	0	1	1				
3	X	X	1	X	X	X	1	1	X	0	0	0	0	1	1	8	8	8	80
4	X	X	X	X	X	0	0	X	X	1	X	X	1	0	0	0	8	0	80
5	1	1	X	X	X	X	1	0	0	0	0	1	X	0	1	6	10	2	100
6	1	X	X	X	1	X	1	0	0	0	0	1	X	0	1	5	10	0	100
7	0	X	1	0	0	X	X	X	1	0	0	0	0	1	1	6	10	0	100
8	0	X	1	0	0	1	X	X	1	0	0	0	0	1	1	6	10	0	100
9	0	0	0	0	0	1	X	X	X	X	1	0	0	1	0	3	10	0	100
10	X	X	X	1	X	0	0	X	X	X	X	X	1	0	0	0	10	0	100

Table 3.2 provides more detailed diagnostic information for circuit c17, where columns correspond to faults and the rows to test patterns whereas the following notation is used: X – no faults, 0 (1) – stuck-at-0 (1) fault detected by the test pattern. Moreover, Table 3.2 contains updated information of tables $\{F(t_j)\}$ and $\{F(T_j)\}$ after pattern 2 has been found as failed.

In the case of bisection of patterns (Figure 3.5a) a secondary diagnostic tree DT is built up by bisection of the reminder set of DPs (from 3 to 10), and starts with the test that contains patterns from 3 to 6. For the proposed method, the rows from 3 to 10 in Table 3.2 are updated. Pattern 3 detects 8 faults (marked in light grey) from the suspected list (marked in dark grey). Pattern 4 and 10 are not needed for building up a secondary DT since they do not detect any suspected fault. Pattern 5 detects 2 suspected faults not yet tested by pattern 3. Patterns 6-9 also not used for a secondary DT since they detect suspected faults that are already covered by patterns 3 and 5. Therefore, according to the proposed algorithm, a secondary search tree DT (Figure 3.5b) consists of only two nodes: 3 and 5, since patterns 3

and 5 together test all the suspected faults and one of them will definitely fail during next test session.

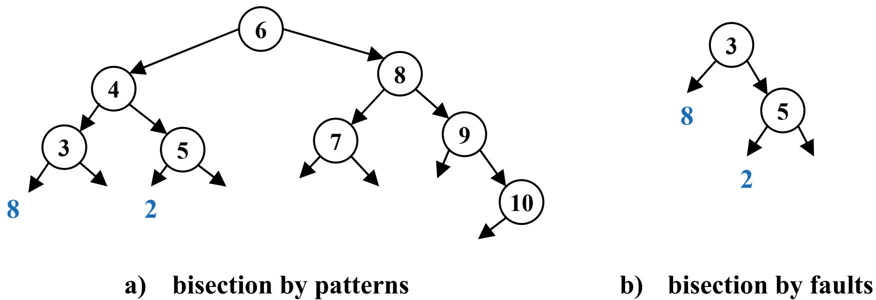


Figure 3.5 *Secondary diagnostic trees for c17*

If now to make a comparison between secondary diagnostic trees of the proposed algorithm and the classical binary search, the difference becomes evident. The tree in Figure 3.5b contains only nodes (test patterns) that are really suspected to fail, while the classical binary search tree contains all nodes. As a consequence, for the secondary tree of the proposed method average number of test sessions is (8×1 + 2×2) / 10 = 1.2 and average number of clocks equals to (3−2)×8 + ((3−2) + + (5−3))×2 = 1.4 compared to classical binary search (3×8 + 3×2) / 10 = 3 and (((6−2) + (4−2) + (3−2))×8 + ((6−2) + (4−2) + (5−4))×2) / 10 = 7, respectively. The superiority of the proposed method consists in supposition that the more failed patterns are found, the less faults are under suspect and as a result secondary trees will contain less nodes and less test sessions will be needed to identify a new failing pattern.

To conclude the description of the proposed algorithm main features are listed below to draw attention to it main concepts.

Main features of the proposed method:

- The diagnostic information inherent in test patterns is taken into account:
  - fast search of the first failing pattern (search tree shifted to the right)
  - allows pruning the search space (patterns that do not detect suspected faults are omitted, not used for building up a search tree)
  - not all failing patterns are necessarily needed to be fixed for diagnosis, i.e. diagnosis is finished when an appropriate diagnostic resolution is achieved

3.2.4 Logical operations needed to support the proposed method

Support of the proposed method in BIST environment can be complicated since there is a need for extra memory to keep dynamically updated fault tables. There is also a need for extra hardware to perform some calculations with fault tables, so as to define new tests and to update a list of suspected faults. However, these calculations are simple logical operations and can be easily implemented in hardware. According to the proposed algorithm, described in section 3.2.2, there are two basic operations employed to perform effective search of failing patterns:

- calculation of total fault vectors
- updating list of suspected faults with failed or passed patterns

3.2.4.1 Calculation of total fault vectors

Calculation of total fault vectors $F(T_j)$ for total fault table $\{F(T_j)\}$ generation is essential since the end pattern t_{end} of a new test is defined by total fault coverage FC_j derived from $\{F(T_j)\}$. The formula for total fault table $\{F(T_j)\}$ generation is presented in the description of the proposed algorithm:

$$\{F(T_j)\}[j] = \{F(t_j)\}[j] \cup \{F(T_j)\}[j\text{-}1]$$

where j denotes the number of test pattern (DP) in the table. Thus, a new total fault vector $F(T_j)$ for a test pattern t_j is calculated by adding the new faults detected by pattern $F(t_j)$ to the faults detected with preceding patterns $F(T_{j\text{-}1})$.

In each cell of a fault table is stored only one of four different values: “X” – no faults, “0” – stuck-at-0, “1” – stuck-at-1 and “&” – both stuck-at faults detected. These values can be coded using 2 bit-code as shown in Table 3.3. Then the required calculation can be performed using simple logical *bitwise OR* operator. The correctness of results obtained with *bitwise OR* operator can be checked against values in Table 3.4, where all possible combinations of operands and results are presented. Indeed, if symbols “X”, “0”, “1” and “&” in operands $F(t_j)$ and $F(T_{j\text{-}1})$ substitute with corresponding codes “00”, “01”, “10” and “11” and apply *bitwise OR* operator, then the same sequence of symbols as in $F(T_j)$ comes out after decoding.

Table 3.3 *Coding the data of fault table by 2-bit code*

Fault	Code
X	**00**
0	**01**
1	**10**
&	**11**

Table 3.4 *Calculation of total fault vectors by bitwise OR operator*

$F(t_j)$	X	X	X	X	0	0	0	1	1	&	a
$F(T_{j\text{-}1})$	X	0	1	&	0	1	&	1	&	&	b
$F(T_j)$	X	0	1	&	0	&	&	1	&	&	$a \cup b$

Moreover, before total fault table generation, when the first failed pattern is already found, the initial fault table $\{F(t_j)\}$ has to be updated by the list of suspected faults F_s :

$$\{F(t_j)\}[j] = \text{initial } \{F(t_j)\}[j] \cap F_s$$

This operation is identical to updating the list of suspected faults F_s by a failed pattern and will be considered in the next section.

3.2.4.2 Updating list of suspected faults

Initially, before diagnosing a failed CUT, all possible SAFs in the CUT are considered as suspected. When diagnostic procedure has started and the first failed pattern has been found (i.e. a leaf of the primary diagnostic tree has been reached), the list of suspected faults is reduced to the number of new faults detected by the failed pattern. If an acceptable diagnostic resolution is not reached, the procedure of searching failing patterns proceeds. Two different outcomes can occur: either a new failed pattern is found or the end of test sequence is reached without identifying a new failing pattern.

Considering the first case, an intersection of suspected faults F_s and faults detected by the new failed pattern $F(t_f)$ has to be found in order to obtain an updated list of suspected faults:

$$F_s = F_s \cap F(t_f)$$

If apply the same coding as for calculation of total fault vectors, then the update of a list of suspected faults with a failing pattern can be accomplished using *bitwise AND* operator as shown in Table 3.5.

Table 3.5 *Updating a list of suspected faults with a failed pattern*

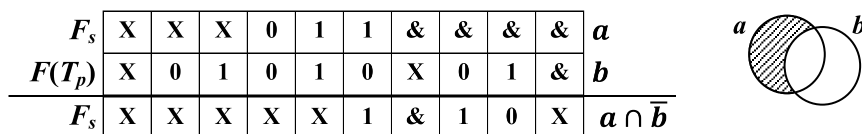
F_s	X	X	X	0	1	1	&	&	&	&	a
$F(t_f)$	X	0	1	0	1	0	X	0	1	&	b
F_s	X	X	X	0	1	X	X	0	1	&	$a \cap b$

Considering the second case, where tests were applied, but a fault was not detected. It means that faults tested by passed patterns $F(T_p) = F(T\,[t_{start}, t_{end}])$ do not cause the circuit to fail. These faults must be removed from the list of suspected faults:

$$F_s = F_s - F(T\,[t_{start}, t_{end}])$$

Moreover, when a new failed pattern is found, there could be a number of passed patterns between two failed patterns. The faults detected by these passed patterns must be also removed from the list of suspected faults. The subtraction can be also performed by *bitwise AND* operator, where inverted values of $F(T_p)$ are employed as shown in Table 3.6.

Table 3.6 *Updating a list of suspected faults with passed pattern(s)*



F_s	X	X	X	0	1	1	&	&	&	&	a
$F(T_p)$	X	0	1	0	1	0	X	0	1	&	b
F_s	X	X	X	X	X	1	&	1	0	X	$a \cap \overline{b}$

In conclusion, it must be mentioned that the operations needed to support the proposed method in BIST environment, are simply implemented in hardware and also very fast to perform in parallel. All required computations are carried out each time when a failed pattern is found and it is needed to perform the same number of operations as the number of remaining patterns in a test sequence. Due to a probable significant number of iterations needed to perform fault diagnosis for a large circuit using the proposed method, the diagnostic procedure can slow down compared to others algorithms that do not use additional computations. On the other hand, the proposed method may save more time running less number of tests and achieving acceptable diagnostic resolution without identifying all failed patterns.

3.2.5 A modification to implementation of the proposed method in BIST environment

While performing fault diagnosis in BIST environment based on the proposed method, test patterns are being selected from a pseudorandom test sequence and are serving as *diagnostic points* (DPs). Diagnostic data of DPs are stored in BIST memory due to significant importance for the diagnostic procedure.

Initially, when a development of the proposed method had started, by default was decided that a test pattern is selected as DP only if it detects new faults, not yet detected by preceding patterns. As a result, the diagnostic data provided by most patterns of the pseudorandom test sequence was unused and the initial sequence of test patterns transformed to significantly smaller one, since only a small number of test patterns were DPs compared to the number of all patterns. Diagnostic data provided only by DPs speeded up the diagnostic procedure since a search tree contained a small number of nodes. Also, when a leaf of the tree was reached (a failed pattern was found), only faults detected by this failed pattern were being suspected. This number of faults was usually quite small since a single pattern could not test a lot of them. Advantages of such selection of DPs were evident, but unfortunately this approach did not meet the demands of the BIST environment.

Consider, as an example, a test sequence of 10 patterns where as DPs are selected patterns 1, 2, 5, 7 and 10 (see Figure 3.6a). Assume that during fault diagnosis of a CUT the first pattern to fail was pattern 2. Since patterns that detect new faults are selected as DPs, then as a first result of diagnosis, all new faults tested by pattern 2 are suspected. According to the proposed algorithm, the procedure of searching new failing patterns must be continued from pattern 5, since it is the next DP. Thus, diagnostic data of patterns 3 and 4 are omitted. For instance, if the new test session contains two DPs, patterns 5 and 7, then its execution in the BIST environment will be complicated. First, pattern 5 has to be loaded into TPG and then applied to inputs of the CUT. After that, TPG must be stopped and pattern 7 should be loaded into TPG and applied as input stimuli. Then a signature obtained with ORA is compared to the fault-free signature stored in BIST memory to ascertain pass/fail of the CUT. Thus, this approach is not taking into account diagnostic data of pattern 6 as well as it needs loading patterns in TPG twice (or simply shifting the values of pattern 6 in TPG registers without loading them into the inputs of the CUT).

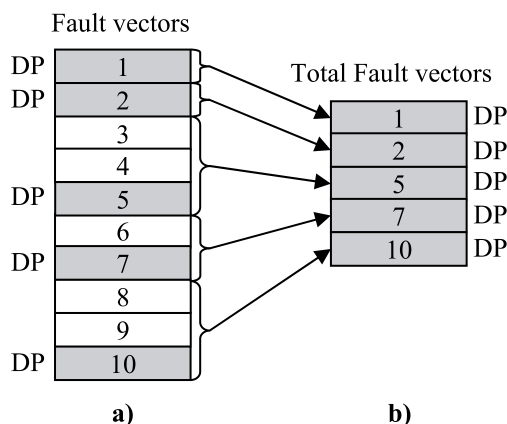


Figure 3.6 *Transformation of initial sequence of fault vectors (a) to a sequence of DPs (b)*

So as to avoid the loss of diagnostic data and simplify the diagnostic procedure in the BIST environment, a following modification is introduced. In diagnostic data of DPs must be stored not only faults detected by the corresponding DP as was before, but also faults detected by a set of patterns located between given DP and the preceding DP as shown in Figure 3.6b. Let go back to the test that contains the same DPs, pattern 5 and 7, as before. Now, when pattern 5 is loaded into TPG the generation of pattern 6 and 7 is done automatically by shifting values in LFSR registers. Moreover, if pattern 7 denoted as DP fails then now faults detected by patterns 6 and 7 are being suspected.

Experimental comparison and analysis of the results between the initial and the modified methods are presented in the next section.

3.3 Experimental data

Experiments were carried out on the ISCAS'85 benchmark circuits [92] [93] using Turbo Tester toolset [94] for generating pseudorandom test patterns and for stuck-at fault simulation.

The general data of the circuits and test sequences are presented in Table 3.7. "*All*" means the full length of the pseudorandom test sequence, *"Eff"* stands for the number of "efficient" patterns which detect new faults not yet tested by previous patterns (only these patterns as DPs are involved in the diagnostic analysis), "*Tested Faults*" means the number of tested stuck-at faults by given test sequence, and "*All Faults*" – all possible stuck-at faults in the circuit. The average and worse diagnostic resolutions (the numbers of suspected faults at the end of diagnostic procedure) are calculated over all possible diagnosis results (over all tested faults in the circuits) that can be achieved for the given circuits by the given pseudorandom test sequences. The best diagnostic resolution for all circuits was 1. For instance, if circuit c432 has failed during the test then as a result of fault diagnosis on average 3.2 faults will be suspected, and 10 faults in the worst case, to be the cause of an error. Also, 616 − 573 = 43 faults in circuit c432 cannot be diagnosed since they are not tested by given test sequence. The main goal of experiments was to compare different algorithms for searching failing patterns and not to generate optimal pseudorandom test sequences that detect all possible stuck-at faults in minimal test sequence length.

Table 3.7 *Diagnostic data for ISCAS'85 circuits*

Circuit	# Patterns		# Faults		Diagnostic resolution	
	All	Eff	Tested	All	Average	Worse
c432	223	65	573	616	3.2	10
c499	1373	100	1194	1202	2.3	5
c880	2692	108	994	994	1.9	10
c1355	1438	113	1610	1618	3.1	5
c1908	4420	175	1723	1732	3.3	16
c2670	22862	116	2328	2626	4.1	45
c3540	9631	249	3149	3296	2.4	28
c5315	1793	214	5364	5424	2.3	20
c6288	42	28	7693	7744	3.3	11
c7552	24337	309	6684	7104	2.7	13

3.3.1 Analysis of results obtained during comparison between the initial method and the modified method

As was mentioned in section 3.2.5, a modification to the diagnostic data of DPs stored in memory must be performed to meet demands of the BIST environment.

Therefore experiments were carried out with the same ISCAS'85 benchmark circuits to find out consequences of the introduced modification.

The obtained results of the comparison are shown in Table 3.8, where best, average and worse diagnostic resolutions achieved as a result of fault diagnosis, as well as minimal, average and maximal number of test sessions needed to find out all failing patterns for both methods are presented. The ratio of average results of the modified method relatively to average results of the initial method is calculated and presented at the bottom of Table 3.8. According to it, the average number of test sessions has been increased by 1.84 times while the average diagnostic resolution has been improved by 1 / 0.95 = 1.05 times. The main reasons for that consist in a fact that in the modified approach diagnostic data of all test patterns are taken into account, not only from effective patterns (DPs) as in the initial approach.

Table 3.8 *Comparison of results between the initial and the modified methods*

Circuit	Method	Diagnostic resolution			Test sessions		
		Best	Average	Worse	Min	Average	Max
c432	Initial	1	3.2	10	6	12.0	45
	Modified	1	2.6	10	6	15.4	54
c499	Initial	1	2.3	5	6	26.2	65
	Modified	1	2.3	5	6	44.0	92
c880	Initial	1	1.9	10	7	17.0	51
	Modified	1	1.8	6	7	27.1	71
c1355	Initial	1	3.1	5	7	27.5	76
	Modified	1	3.1	6	7	43.9	100
c1908	Initial	1	3.3	16	7	31.2	124
	Modified	1	2.9	16	7	62.8	168
c2670	Initial	1	4.1	45	5	20.6	120
	Modified	1	4.1	45	5	37.2	120
c3540	Initial	1	2.4	28	6	27.0	148
	Modified	1	2.2	28	6	53.9	203
c5315	Initial	1	2.3	20	8	25.9	129
	Modified	1	2.2	13	8	45.4	170
c6288	Initial	1	3.3	11	9	15.9	25
	Modified	1	3.3	8	9	17.4	26
c7552	Initial	1	2.7	13	8	42.8	214
	Modified	1	2.6	14	8	105.3	260
Ratio	**Initial**	**1**	**1**	**1**	**1**	**1**	**1**
	Modified	**1.00**	**0.95**	**0.93**	**1.00**	**1.84**	**1.27**

For instance, assume that in Figure 3.6 only patterns 5 and 6 can detect a fault, where pattern 5 is selected as DP. Both methods will find out that pattern 5 fails. But the initial approach will suspect only faults detected by pattern 5, while the modified method will also suspect faults detected by patterns 3 and 4.

In the initial approach the result of diagnosis D_I will be following:

$$D_I = f_5 - (f_1 \cup f_2 \cup f_7 \cup f_{10})$$

where f_j means faults tested by pattern *j*. In the modified approach a test session containing the DP 7 (patterns 6 and 7) will also fail because of pattern 6 and a result of diagnosis D_M will be different:

$$D_M = (f_3 \cup f_4 \cup f_5) \cap (f_6 \cup f_7) - (f_1 \cup f_2 \cup f_8 \cup f_9 \cup f_{10})$$

For the modified method additional test session is needed to identify DP 7 as failing. Therefore average number of test sessions in the modified method is higher. On the other hand, the diagnostic resolution has been slightly improved, because tests patterns that are not selected as DPs now assist in reducing the list of suspected faults.

3.3.2 Comparison of searching algorithms

Comparison of the four algorithms: classical *Binary Search*, *Doubling*, *Jumping* and the proposed algorithm in numbers of test sessions needed for fault diagnosis is presented in Table 3.9. Minimum, average and maximum numbers of test sessions over all possible diagnosis results for the given circuits at the given pseudorandom test sequences are presented, where a search of failing patterns was carried out only among the set of effective patterns selected as DPs (see Table 3.7).

Table 3.9 *Comparison of fault diagnosis methods*

Circuit	Binary			Doubling			Jumping			Proposed		
	min	avrg	max	min	avrg	max	min	avrg	max	min	avrg	max
c432	6	29.0	201	5	26.3	80	7	24.1	120	6	12.0	45
c499	7	106.0	342	7	66.3	121	7	70.7	181	6	26.2	65
c880	6	56.9	289	6	42.3	117	7	41.1	143	7	17.0	51
c1355	7	113.4	419	10	72.2	140	9	76.2	214	7	27.5	76
c1908	7	149.5	757	8	84.9	208	6	89.6	351	7	31.2	124
c2670	6	76.0	686	7	52.3	146	6	52.4	345	5	20.6	120
c3540	7	126.3	990	8	80.1	293	7	78.8	436	6	27.0	148
c5315	7	109.8	819	6	79.0	252	7	75.6	368	8	25.9	129
c6288	5	33.9	69	7	26.2	38	7	28.1	54	9	15.9	25
c7552	8	243.6	1497	7	138.9	368	8	141.9	616	8	42.8	214
Average	**6.60**	**104.44**	**606.90**	**7.10**	**66.85**	**176.30**	**7.10**	**67.85**	**282.80**	**6.90**	**24.61**	**99.70**

The proposed method outperforms all other methods. According to Table 3.10 which is based on results presented in Table 3.9, the average length of test sessions, for the proposed method is 4.24 times shorter than for the classical *Binary Search*, 2.72 times shorter than for the *Doubling* and 2.76 times shorter than for *Jumping*

algorithms. Regarding the worst (longest) test session, the proposed method outperforms the *Binary Search* by 6.09 times, *Doubling* by 1.77 times and *Jumping* by 2.84 times. All the algorithms give rather similar results when comparing the shortest test sessions.

Table 3.10 *Comparison of average test lengths*

Algorithms	Ratio of test sessions		
	MIN	AVRG	MAX
Proposed	**1**	**1**	**1**
Binary	0.96	4.24	6.09
Doubling	1.03	2.72	1.77
Jumping	1.03	2.76	2.84

Since the goal of all the algorithms in given experiments was to find out all the failing patterns then the achieved diagnostic resolutions for all the algorithms are the same as presented in Table 3.7. In Figure 3.7 the diagnostic resolutions for circuit c2670 over all stuck-at faults that were possible to diagnose by given test sequence are depicted. The figure helps easily to identify large blocks of indistinguishable faults. For given case, there are blocks of size 25, 29 and 45 faults whereas for most faults the diagnostic resolution is between 1 and 5. For instance, for 664 of faults the diagnostic resolution is 2 according to Figure 3.7. If the achieved diagnostic resolution is not acceptable for all the faults, then either additional DPs have to be selected from the test sequence or extra test patterns have to be generated in order to split large blocks of indistinguishable faults into smaller ones.

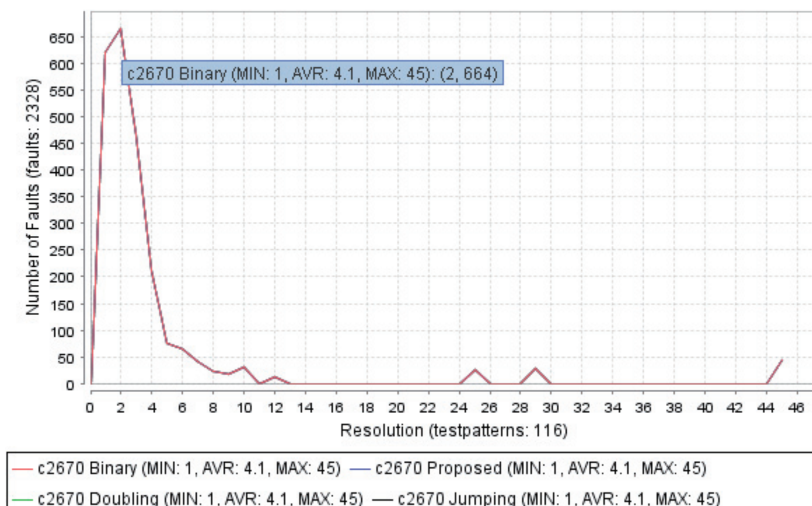


Figure 3.7 *Diagnostic resolutions for c2670*

3.4 Conclusions

A new method is proposed for embedded fault diagnosis in digital systems with BIST environments. Compared to the classical bisection of test pattern sets, in this thesis two novelties have been introduced:

- instead of test patterns as in the classical *Binary Search*, the detected fault sets are the objectives of bisection which allows to reduce the average length of the diagnostic procedure
- due to the sequential character of the new method the set of suspected faults during the diagnostic procedure is continuously updated in a way which allows to prune the search space and to exclude the need of finding all the failed test patterns as in the case of the classical *Binary Search*, which additionally increases the speed of diagnosis

The proposed method is compared with three known fault diagnosis methods: classical *Binary Search*, *Doubling* and *Jumping*. Experimental results demonstrate that the new method outperforms considerably the others methods. The average length of test sessions for the proposed method is 4.2 times shorter than for the classical *Binary Search*, and about 2.7 times shorter than for the *Doubling* and *Jumping* algorithms.

Chapter 4

METHODS FOR INCREASING DIAGNOSTIC RESOLUTION

In this chapter a method to improve fault diagnosis accuracy in terms of fault resolution in digital systems using *built-in self-test* (BIST) facilities is proposed. The diagnostic resolution improvement is attained by partitioning a single signature analyzer into a set of multiple independent analyzers. The algorithms are given to synthesize an optimal interface between the outputs of the circuit under test and the signature analyzers. Experimental results demonstrate the feasibility and efficiency of the approach.

4.1 Fault diagnosis challenges in BIST environment

In Chapter 3 a modification of the *Binary Search* driven by bisection of fault coverage instead of the classical bisection of patterns [83] [84] [85] [86] has been proposed. It has been already shown that the new method outperforms the known algorithms in the average length of the diagnostic procedure. However, there is still a motivation for further investigations because the pseudorandom essence of BIST is not providing high diagnostic resolutions.

Three directions in development of the proposed approach can be considered:

- improvement of diagnostic resolution
- decrease in number of used *diagnostic points* (DPs)
- decrease in number of test sessions needed for diagnosis

In Table 3.7, the last column “worse diagnostic resolution” shows the largest blocks of indistinguishable faults, as the result of fault diagnosis. The largest sizes of blocks are obtained for circuits c2670 and c3540. The achieved diagnostic resolution 45 and 28 faults, respectively, is not acceptable and must be improved. It may be attained by selecting additional DPs that split large blocks of indistinguishable faults into smaller ones. It is easy to find them out by looking up the fault table where diagnostic data for all test patterns are stored. Thus, the number of DPs needed for fault diagnosis will grow with diagnostic resolution improvement.

On the other hand, solving the first challenge aggravates the second one, decrease in number of used DPs. Hence, there is need for an efficient algorithm that will be able to select minimal number of DPs to achieve highest diagnostic resolution. Selecting minimal number of DPs is essential for embedded fault diagnosis since it saves the memory needed to store diagnostic data of DPs and also speeds up diagnostic application time since fewer nodes are used in building up a search tree. Considering complex circuits, where long sequences of pseudorandom test patterns are applied to test a huge number of possible stuck-at faults in a *circuit under test* (CUT), the selection of the optimal DPs will be extremely time-consuming. The selected DPs will be very efficient only for given test sequences.

The third challenge consists in decreasing a number of test sessions in order to augment the superiority of the proposed method compared to others methods considered in Chapter 3. It could be attained by searching and processing only a part of all failed patterns to reach still acceptable resolution, as was mentioned earlier.

In this chapter a further improvement of the diagnostic procedure by using multiple signature analyzers will be presented. The approach attempts to solve both fault diagnosis challenges in BIST environment: improvement of diagnostic resolution and decrease in number of test sessions.

4.2 BIST with multiple signature analyzers

Assume a CUT with a set of faults F has n outputs where each output i may be influenced by a subset of faults $F_i \subseteq F$. Introduce m, $1 < m \leq n$, *signature analyzers* (SA) which should be connected to the outputs of the CUT. An example of such a BIST for fault diagnosis with three SAs is shown in Figure 4.1.

Denote by I_j the set of outputs of the CUT connected to the signature analyzer SA_j. Depending on the faults detected by the outputs I_j, each SA_j may be influenced by the following subset of faults:

$$S_j = \bigcup_{i \in I_j} F_i$$

In other words, if there is a fault $f \in F$ in the CUT, this fault will be detected by all signature analyzers SA_j where $f \in S_j$. As an example, the fault in the CUT highlighted in Figure 4.1 can be detected via two output lines by SA_1 and SA_2 .

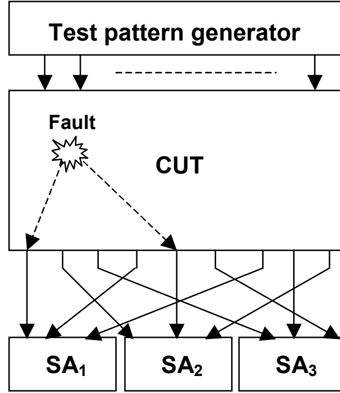


Figure 4.1 *BIST with multiple signature analyzers*

Introduce for a set of m signature analyzers a codeword C_k as a sequence of bits $C_k = (c_m , c_{m-1} , \dots , c_1)$, so that the index k represents the decimal value of the binary codeword C_k . Represent by C_k the result of testing, so that $c_j = 1$ when the signature analyzer SA_j has detected a fault, and otherwise, $c_j = 0$ when no faults has been detected by SA_j .

The case when no faults has been detected by a set of signature analyzers $\{SA_m , SA_{m-1} , \dots, SA_1\}$ corresponds to the codeword C_0 . For any other codeword C_k , $k \neq 0$, a diagnosis can be stated as a subset of suspected faults:

$$D_k = F^1 - F^0 = \bigcap_{j:c_j=1} S_j - \bigcup_{j:c_j=0} S_j \subset F$$

where F^1 is the intersection of subsets S_j of faults tested by SAs with erroneous signatures ($c_j = 1$), and F^0 is the union of subsets S_j of faults tested by SAs with correct signatures ($c_j = 0$). It is evident that for all $l \neq k$, $1 \leq l,k \leq 2^m - 1$, $D_l \cap D_k = \emptyset$, and

$$\bigcup_{k=1,2,\dots,2^m-1} D_k = F$$

In Figure 4.2, seven intersections of fault sets are shown to illustrate the fault diagnosis by three signature analyzers. For example, if a fault is detected by analyzers SA_1 and SA_2 , the codeword $C_3 = (011)$ will be produced, which corresponds to the subset of suspected faults:

$$D_3 = S_1 \cap S_2 - S_3$$

as the result of diagnosis.

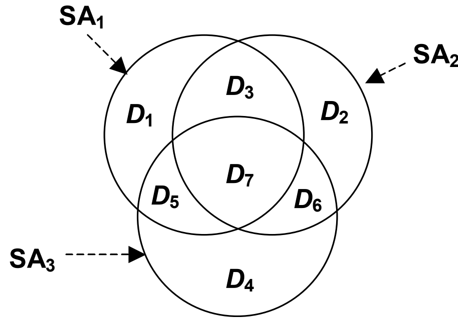


Figure 4.2 *Diagnosis by the set of three signature analyzers*

It is evident that the best diagnostic resolution will be achieved when the suspected subset of faults for any result of diagnosis, i.e. for any codeword C_k, will be minimal. From this statement the following task can be formulated to find the best interface between the outputs of CUT and the set of signature analyzers by connecting the SAs to CUT in such a way that:

$$\forall k,\ k = 1, 2, \ldots, 2^m - 1: \left[\frac{|F|}{2^m-1}\right] \leq |D_k| < \left[\frac{|F|}{2^m-1}\right] + 1 \qquad (4.1)$$

Here $[x]$ denotes the largest integer that is less than or equal x. In the ideal case, provided that $\frac{|F|}{2^m-1}$ is integer, the situation should be reached where

$$D_1 = D_2 = \ldots = D_{2^m-1} = \frac{|F|}{2^m - 1}$$

4.3 Design of the interface between CUT and SAs

Algorithms of designing the best interface between CUT and the set of SAs has been developed so that the condition (4.1) is as closely as possible satisfied.

4.3.1 Algorithm "Equal Subsets"

In the *"Equal Subsets"* algorithm an interface is constructed as a procedure where the outputs of a CUT are assigned to SAs step by step in such a way that in each step the condition (4.1) is satisfied as closely as possible.

To give the words "as closely as possible" a countable meaning, the following notions are introduced:

- ideal size of the set D_k measured as $D_{ideal} = \frac{|F|}{2^m-1}$
- distance of D_k from the ideal size measured as $\Delta_k = D_{ideal} - |D_k|$

In the ideal case the fault resolution, i.e. the number of suspected faults is for all faults equal and minimal. Practically, the situation should be strived where the average number of suspected faults for all faults will be minimal, i.e. the average of Δ_k should be minimized.

The whole procedure of designing the interface consists of two parts. In the first part (*Algorithm 4.1*), to each SA an initial output of CUT is assigned. In the second part (*Algorithm 4.2*), on each step an arbitrary output of CUT is selected, and a SA is found as the best solution to be connected to the selected output of CUT. Consider only the practical situations where the number of outputs n of CUT is much greater than the number m of SAs, $m << n$ and $n >> 2$. In the extreme case of $m = n$, each output of CUT is separately observable, and no SAs is needed.

Algorithm 4.1:

1. Order the set of outputs OUT of CUT so that $|F_i| \geq |F_{i+1}| \geq |F_{i+2}| \geq \ldots$, $i \in [1, n]$. Take all $S_j = \emptyset$. Take $j = 1$.
2. Take the first $| F_i |$, $i = 1$.
3. Assign i to SA_j, $S_j = S_j \cup F_i$. Remove i from OUT.
4. Modify $j = j + 1$.Take the next SA_j.
5. Calculate for all $i \in OUT$:

BEGIN

$$F(i) = F_i \cup \bigcup_{l=1}^{j} S_l\,, \quad D_{i,ideal} = \frac{|F(i)|}{2^j - 1}$$

Calculate for all C_k, $k = 1, 2, ..., 2^j - 1$:

$$D_k = \bigcap_{j:c_j \in C_k,\, c_j=1} S_j - \bigcup_{j:c_j \in C_k,\, c_j=0} S_j \;;\; \delta_k = D_{i,ideal} - |D_k|$$

$$\Delta_i = \sum_{k=1}^{2^j-1} \delta_k$$

END

6. Find i^*, so that $\Delta_{i^*} = \min \Delta_i$, where $i \in OUT$.
7. Assign i^* to SA_j, $S_j = S_j \cup F_{i^*}$. Remove i^* from OUT.
8. If $j < m$, go to 4, otherwise END.

To engage right in the beginning into the intersection procedure as many fault as possible, it is reasonable to start the *Algorithm 4.1* with assigning to the first SA the output of CUT with the largest set of detected faults (Steps 1-3). In Step 4, as a current solution j outputs have been assigned to j different SAs, so that the average distance Δ_k from the ideal diagnostic resolution is minimal. In Steps 5-7 the next output to be assigned to the next SA is chosen, so that the average distance Δ_k from the ideal diagnostic resolution will be minimal. The algorithm is finished when to all SAs a single output of CUT is assigned (connected).

The goal of the *Algorithm 4.2* is to assign the remaining outputs to SAs in the way that the average distance Δ_k from the ideal resolution will be minimal to reach the best resolution for all of faults in CUT.

Algorithm 4.2:

1. Take the next i from OUT.
2. Calculate: $F(i) = F_i \cup \bigcup_{j=1}^{m} S_j$; $D_{i,ideal} = \frac{|F(i)|}{2^m - 1}$
3. Calculate for all S_j :

   BEGIN

   $S_j = S_j \cup F_i$

   Calculate for all C_k, $k = 1, 2, ..., 2^m - 1$:

   $$D_k = \bigcap_{j:c_j \in C_k,\, c_j=1} S_j - \bigcup_{j:c_j \in C_k,\, c_j=0} S_j \ ; \quad \delta_k = D_{i,ideal} - |D_k|$$

   $$\Delta_j = \sum_{k=1}^{2^m-1} \delta_k$$

   Restore initial S_j

   END
4. Find j^*, so that $\Delta_{j^*} = \min \Delta_j$, $j = 1, 2, ..., m$
5. Assign i to SA_{j^*}, $S_{j^*} = S_{j^*} \cup F_i$
6. Remove i from OUT.
7. If $OUT \neq \emptyset$, go to 1, otherwise END.

Differently from *Algorithm 4.1* where a selection of an item was made from the set of outputs of CUT for a given SA, in *Algorithm 4.2*, a selection is made from the set of SAs for a given output of CUT. In Step 2, a set of all faults $F(i) \subseteq F$ will be calculated which are detected by the outputs already connected to SAs and by the output selected in Step 1 for connection. In Step 3, the average distance Δ_j from the ideal diagnostic resolution for all the SAs is calculated with assumption that the selected output i is already connected to SA_j . In Steps 4-6 the best connection between the output i and the SAs is decided. The *Algorithm 4.2* is finished when all the outputs of CUT are connected to SAs.

The *Algorithms 4.1* and *4.2* are targeting the optimum interface to achieve the highest diagnostic resolution when diagnosing a failing circuit. The algorithms are based on the greedy technique.

A number of experiments were performed to approve effectiveness of the algorithm "*Equal Subsets*". The obtained results (see section 4.5.1) and also deeper analysis of circuits in Table 4.1 give an understanding that the ideal case when

$$D_1 = D_2 = \ ... \ = D_{2^m-1} = \frac{|F|}{2^m - 1}$$

is practically impossible to achieve when constantly increasing the number of signature analyzers. The cause of it is that most faults can be detected only by single or several outputs. In Table 4.1 faults detection statistics by outputs for ISCAS'85 circuits [92] [93] are presented. "# faults" means number of possible stuck-at faults, "# outs" – number of outputs for given circuits. Columns "1-10" contain percentage of stuck-at faults that can be detected with given number of outputs. For instance, in circuit c1908 46.3% of faults are detected only by a single output. "sum %" denotes the sum of values from columns "1-10". For circuit c1908, each of 100% − 57.7% = 42.3% of all faults can be detected by more than 10 outputs whereas the circuit has only 25 outputs.

Table 4.1 *Faults detection statistics for ISCAS'85 circuits*

circuit	# faults	# outs	# faults detected by given number of outputs (%)										sum %
			1	2	3	4	5	6	7	8	9	10	
c432	616	7	13.3	4.2	4.6	10.1	23.7	29.6	14.6	0	0	0	100
c499	1202	32	32.0	0	0	6.7	0	0	0	0	0	0	38.6
c880	994	26	55.7	6.6	3.0	3.0	5.0	5.0	2.8	5.0	9.1	2.4	97.8
c1355	1618	32	31.6	0	0	4.9	0	0	0	0	0	0	36.6
c1908	1732	25	46.3	0	0	0	0	0.4	1.9	9.2	0	0	57.7
c2670	2626	140	15.0	15.8	31.6	15.0	1.8	4.4	4.7	2.7	0	2.8	93.8
c3540	3296	22	12.9	1.9	2.6	0	32.3	6.3	3.3	2.4	2.3	10.0	74.0
c5315	5424	123	28.8	9.7	0.2	0.7	30.6	16.3	2.4	0.2	0.9	0.3	90.2
c6288	7744	32	3.2	0.4	0.8	1.2	1.6	2.0	2.4	2.8	3.3	3.7	21.3
c7552	7104	108	39.3	17.5	6.7	7.7	6.0	6.8	0.2	0.2	0.5	0.5	85.4
Average			**27.80**	**5.61**	**4.95**	**4.93**	**10.11**	**7.08**	**3.23**	**2.26**	**1.61**	**1.97**	

According to Table 4.1 a reasonable decision can be taken about the number of SAs to be used in fault diagnosis, so as to get the most efficient improvement in average diagnostic resolution that could be up to $2^m - 1$ times better than initial one, where m denotes the number of employed SAs.

4.3.2 Algorithm "Unique Faults"

Based on information obtained in Table 4.1 and also on an assumption that quite large amount of faults are detected by a single output, the algorithm "*Unique Faults*" has been introduced that attempts to assign outputs of circuits to SAs in a different way than the "Equal Subsets" algorithm.

Algorithm 4.3:

1. Order the set of outputs *OUT* of CUT so that $|F^*_i| \geq |F^*_{i+1}| \geq |F^*_{i+2}|$, $i \in [1, n]$, where $|F^*_i|$ – number of "*unique*" faults
2. Take all $S_j = \varnothing$.
3. Assign single i to each SA_j from ordered set of outputs *OUT*. $S_j = S_j \cup F^*_i$. Remove i from *OUT*.
4. Take the next i from *OUT*
5. Calculate: $F^*(i) = F^*_i \cup \bigcup_{j=1}^{m} S_j$; $D^*_{i,ideal} = \frac{|F^*(i)|}{m}$
6. Calculate for all S_j:

   BEGIN

   $S_j = S_j \cup F^*_i$

   Calculate for all S_j: $\delta_j = D^*_{i,ideal} - |D^*_j|$

   $$\Delta_j = \sum_{j=1}^{m} \delta_j$$

   Restore initial S_j

   END
7. Find j^*, so that $\Delta_{j^*} = \min \Delta_j$, $j = 1, 2, ..., m$
8. Assign i to SA_{j^*}, $S_{j^*} = S_{j^*} \cup F^*_i$
9. Remove i from OUT.
10. If $OUT \neq \varnothing$, go to 4, otherwise END.

Firstly, in Step 1 the *Algorithm 4.3* finds out "*unique*" set of faults F^*_i for each output i, i.e. faults detected only by given output, and sorts them in decreasing order. In Steps 2 and 3, a single output i is assigned to each SA_j from the ordered set of outputs *OUT*. Then, in Steps 4-9, the next output from the sorted set is taken and assigned to an analyzer in such way that the number of "*unique"* faults would be equally distributed between analyzers. The *Algorithm 4.3* is finished when all outputs have been assigned to SAs. It takes less time to run, since most computations are made at each iteration of assigning a new output to SA, and the algorithm "*Unique Faults*" calculates only m times the deviation δ from the ideal case compared to $2^m - 1$ times as for the algorithm "*Equal Subsets*", where m – number of SAs. The difference is slight when employing only a few SAs, but with a large number of SAs it becomes significant. For m equal to 10, the algorithm "*Unique Faults*" runs approximately $(2^{10} - 1) / 10 = 102.3$ times faster than the "*Equal Subsets*" algorithm. The effectiveness of the considered algorithms is compared in the section 4.5.1.

4.4 Fault diagnosis with a set of SAs

The fault diagnosis algorithm is a generalization of the algorithm proposed in previous chapter that is based on bisection by fault coverage, but for the case of a set of m signature analyzers. To implement the algorithm in BIST environment a table of diagnostic data is created that consists of:

- the numbers j of test patterns t_j selected as DPs
- signatures $s_k(t_j)$ corresponding to the contents of SA_k , $k = 1, 2, ..., m$, if the test pattern t_j would be the final pattern of the current diagnostic test session
- the sets of faults $F_k(t_j)$ detected by SA_k after the test patterns t_j

For each test pattern t_j the set of faults $F_k(T_j)$ detected in SA_k by the test sequence T_j with the final pattern t_j is calculated:

$$F_k(T_j) = \bigcup_{t_j \in T_j} F_k(t_j)$$

The cumulative fault coverage reached by the test sequence T_j with final pattern t_j is also calculated:

$$FC_j = \bigcup_{k=1}^{m} F_k(T_j) \ / \ |F|$$

where F is the set of all possible faults detected by all the SAs and the whole pseudorandom test sequence.

Consider as an example in Figure 4.3 a table where the rows correspond to the test patterns selected as DPs from pseudorandom test sequence, and the contents of rows illustrate the codewords as the results of testing for the case of $m = 3$. The column "*No*" refers to the number of the test pattern, and the column "*Diagnosis*" refers to the sets of suspected faults found at the given test pattern.

In the first step the whole test sequence is carried out with the last test pattern v. In this step according to the codeword $C_7 = 111$ as the result of test all the SAs fail and the following diagnosis can be made $D_1 = R_{1'} \cap R_{2'} \cap R_{3'}$. To improve the diagnostic resolution according to the bisection algorithm, the partial test sequence up to the test pattern k is carried out. A possibly improved diagnostic resolution results: $D_2 = R_{1''} \cap R_{2''} - R_{3''}$, where $|D_2| \leq |D_1|$. Further diagnostic resolution by reducing can be achieved by continuing the bisection algorithm using the results of only SA_3. The best resolution will be obtained at the test pattern l where the SA_3 fails for the first time. The result of fault diagnosis will be: $|R_3| \leq |R_{3''}| \leq |R_{3'}|$.

Now the fault diagnosis is continued by bisection algorithm based on the results of SA_1 and SA_2. A test sequence with the final pattern *i* is carried out. Since SA_2 does not fail, the test is continued until the pattern *j* is found where SA_2 the first time fails. As the result, the subset of suspected faults $|R_2| \leq |R_{2''}|$ is obtained. Then the algorithm is continued based on the results of SA_1 only, until the pattern *h* is found where SA_1 the first time fails. The final diagnosis will be $D_3 = R_1 \cap R_2 \cap R_3$ where $|D_3| \leq |D_2| \leq |D_1|$.

No	Codeword			Diagnosis
	SA_3	SA_2	SA_1	
h	0	0	1	R_1
i	0	0	1	$R_{1'''}$
j	0	1	1	R_2
k	0	1	1	$R_{1''}, R_{2''}$
l	1	1	1	R_3
v	1	1	1	$R_{1'}, R_{2'}, R_{3'}$

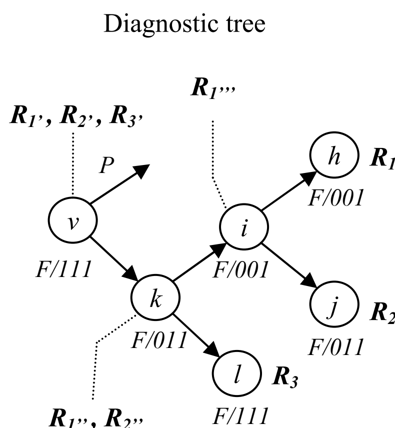


Figure 4.3 *An example of fault diagnosis with a set of three signature analyzers*

4.5 Experimental data

Experiments were carried out on the same ISCAS'85 benchmark circuits [92] [93], applying the same sequences of pseudorandom test patterns and simulating the same number of stuck-at faults as in Chapter 3 (see Table 3.7). The goal of experiments was to prove efficiency of exploiting multiple signature analyzers for diagnostic resolution improvement and also to reduce diagnosis application time by preventing the search of remaining failing patterns when applicable diagnostic resolution has been achieved.

4.5.1 Algorithms of designing the interface between CUT and SAs

In Table 4.2 the results of two different algorithms that design the interface between the CUT and the SAs are presented. In column 3, "ES" means the "Equal Subsets" algorithm and "UF" stands for the "Unique Faults" algorithm. Further, the average diagnostic resolutions over all possible diagnosis results are presented where different number of SAs (1-5) was applied for test responses analysis, and the diagnostic procedure was stopped when either only the first failing pattern

(“1 t_f”) had been found or all failing patterns (“all t_f”) had been detected. In given experiments for the fault diagnosis only effective patterns selected as DPs were used, in which the diagnostic data of remaining patterns were also stored, as described in section 3.2.5. Thus, the average diagnostic resolution for single SA as a result of fault diagnosis, where all the failing patterns were detected, corresponds to the data in Table 3.8 for the modified method.

Table 4.2 *Comparison of algorithms designing the interface between CUT and SAs*

circuit	# outs	Algo-rithms	Average diagnostic resolution at given number of SAs (1-5)									
			1		**2**		**3**		**4**		**5**	
			1t_f	**all t_f**	1t_f	**all t_f**	1t_f	**all t_f**	1t_f	**all t_f**	1t_f	**all t_f**
c432	7	ES	21.3	**2.6**	10.6	**2.1**	7.8	**2.0**	6.1	**2.0**	5.8	**2.0**
		UF			10.8	**2.0**	8.0	**2.0**	7.0	**2.0**	5.8	**2.0**
c499	32	ES	80.2	**2.3**	71.6	**2.0**	72.8	**2.0**	71.2	**2.0**	62.3	**2.0**
		UF			54.3	**2.0**	46.0	**2.0**	44.6	**2.0**	41.7	**2.0**
c880	26	ES	51.7	**1.8**	24.2	**1.7**	17.6	**1.7**	12.7	**1.7**	11.6	**1.7**
		UF			22.7	**1.7**	16.3	**1.7**	12.8	**1.7**	10.4	**1.7**
c1355	32	ES	50.9	**3.1**	46.7	**2.9**	47.6	**2.9**	46.5	**2.9**	45.6	**2.9**
		UF			34.2	**2.9**	29.4	**2.9**	27.3	**2.9**	26.3	**2.9**
c1908	25	ES	99.0	**2.9**	47.1	**2.4**	29.2	**2.4**	20.7	**2.4**	41.0	**2.5**
		UF			45.9	**2.3**	36.1	**2.3**	22.6	**2.3**	17.6	**2.3**
c2670	140	ES	151.5	**4.1**	73.6	**4.0**	43.9	**3.4**	34.9	**3.3**	27.3	**3.3**
		UF			65.2	**3.9**	39.9	**3.3**	30.9	**3.3**	27.5	**3.3**
c3540	22	ES	104.6	**2.2**	45.1	**2.2**	24.3	**2.1**	21.3	**2.1**	15.6	**2.0**
		UF			43.3	**2.1**	27.8	**2.1**	17.5	**2.0**	17.6	**2.0**
c5315	123	ES	232.8	**2.2**	92.0	**2.0**	55.7	**2.0**	41.0	**2.0**	40.7	**1.9**
		UF			87.1	**2.0**	54.2	**2.0**	36.5	**1.9**	28.8	**1.9**
c6288	32	ES	1206.1	**3.3**	475.5	**2.6**	295.5	**2.2**	162.7	**1.9**	101.1	**1.8**
		UF			414.5	**1.9**	214.5	**1.8**	137.0	**1.8**	97.9	**1.8**
c7552	108	ES	262.4	**2.6**	106.8	**2.2**	95.2	**2.1**	65.6	**2.1**	47.9	**2.1**
		UF			103.4	**2.2**	66.9	**2.1**	53.4	**2.1**	45.1	**2.0**
Average		**ES**	**226.1**	**2.7**	**99.3**	**2.4**	**69.0**	**2.3**	**48.3**	**2.2**	**39.9**	**2.2**
		UF			**88.1**	**2.3**	**53.9**	**2.2**	**39.0**	**2.2**	**31.9**	**2.2**

If analyze results in columns “all t_f”, then it is obvious that in most cases either both algorithms give the same results or the “UF” is slightly better. Since all the failing patterns are used for fault diagnosis and average diagnostic resolution for single SA is very high and varies only from 1.8 to 4.1 for all considered circuits, then an employment of multiple SAs is unreasonable. The improvement of the average diagnostic resolution is insignificant. For circuit c2670, exploiting of two SAs instead of single enhances the average diagnostic resolution from 4.1 only to 3.9 for the “UF” algorithm.

The number of used SAs becomes relevant for fault diagnosis when the diagnostic procedure is stopped after the first failed pattern is found. The best average diagnostic resolution improvement is gained for circuit c6288 by 12.3 times, from 1206.1 for single SA to 97.9 for the set of 5 SAs applying the “UF”

algorithm. The obtained result is due to more or less equal distribution of faults between different numbers of outputs as shown in Table 4.1. However, in the ideal case the improvement could be by $2^5 - 1 = 31$ times when exploiting 5 SAs. One of the worst results is obtained for circuit c1355, where the resolution is improved only by 50.9 / 26.3 = 1.9 times when using 5 SAs. This is also caused by structure of the circuit since 31.6% of faults are detected by single output and almost each of remaining ones can be detected by more than 10 outputs as shown in Table 4.1. This makes the distribution of 32 outputs between 5 SAs inefficient since there is a high probability that for most faults all the 5 SAs will fail and there will be no possibility to distinguish faults.

The results for all the considered circuits and numbers of SAs presented in row "Average" show that the "UF" algorithm outperforms the "ES" algorithm. In some certain cases, when analyzing results in columns "1 t_f", the "ES" algorithm achieves better resolution. For instance, results for circuit c1908 when were used 2 SAs are 29.2 for the "ES" and 36.1 for the "UF" algorithms. However, sometimes the "ES" makes the resolution even worse when increasing the number of SAs. For example, considering the same circuit c1908 the resolution for 4 SAs was 20.7 whereas for 5 SAs it became 41.0. Thus, the preference is given to the "UF" algorithm since it designs the interface between CUT and SAs more efficiently, and it is less time-consuming than the "ES" algorithm.

4.5.2 Comparison of different fault diagnosis methods

In this section following algorithms are compared: classical *Binary search* (6) [83] [84] [85] [86], *doubling* (4) [90], *jumping* (5) [91], the proposed method (3) of *bisection detected faults*, the proposed method (2) based on multiple SAs (Mult_SA) where five analyzers were used, and the proposed method (1) where after detecting 5 failed patterns the procedure stopped (Mult_SA_short). For the methods (2)-(6) all failed patterns were used for diagnosis. For experiments were taken the ISCAS'85 circuits c2670, c5315 and c7552 since they have large number of outputs and faults are quite equally distributed between different numbers of outputs which must enhance the efficiency of using multiple SAs. The average results for these three circuits over all faults are depicted in Table 4.3.

A considerable improvement in the speed of diagnosis can be observed for the proposed method of bisection faults (3) compared to the previous methods (4), (5) and (6). The methods (2)-(6) made the diagnosis on the basis of all failing patterns. This is the reason why the diagnostic resolution is equal for all of the methods. Using multiple signature analyzers (2) did not reduce much the test length 53 compared to the proposed method (3) of bisection faults with test length 63. The reason was that all the failed test patterns were targeted. Also the multiple SAs did not have any impact on the diagnostic resolution because of finding all the failing patterns already provided all the needed information which was enough to achieve the best fault resolution at the given test sequence.

Table 4.3 *Comparison of different methods*

Method	Number of test sessions		Diagnostic resolution	
	Average	Max	Average	Worse
(1) Mult_SA_short	12	18	2.7	22
(2) Mult_SA	53	182	2.4	22
(3) Bisect_faults	63	183	2.4	22
(4) Doubling	90	255	2.4	22
(5) Jumping	90	443	2.4	22
(6) Bisect_patterns	143	1001	2.4	22

The real dramatic impact of the multiple SAs emerged when was allowed to stop the procedure at the reduced number of failed test patterns (Mult_SA_short). In the experiment the procedure was stopped after finding 5 failed patterns. The average test length 12 of the diagnostic procedure decreased now in average by 5 times compared to the method (3) with test length 63 while the diagnostic resolution achieved was almost the same as for other methods. The strategy of Mult_SA_short is efficient because it exploits the improved resolution effect of multiple SAs already at the beginning stage of the procedure, which allows to stop it after detecting only a small fraction of failing patterns (Figure 4.4). As the result, a dramatic decrease of the test length as shown in Table 4.3 was achieved.

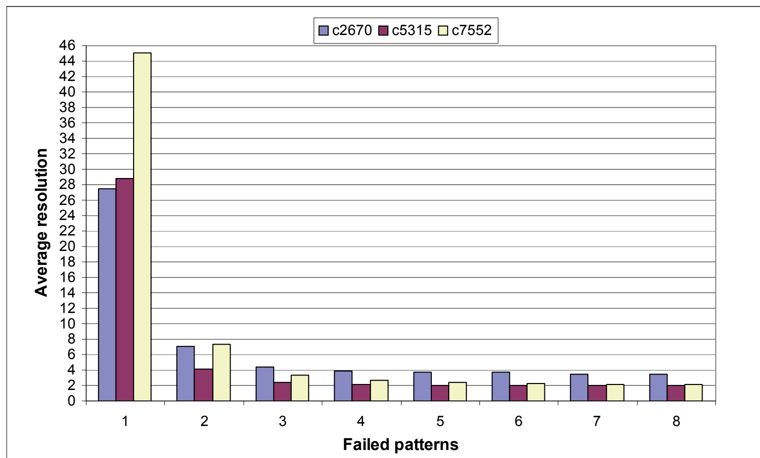


Figure 4.4 *Dependency of the resolution on the test length*

How quickly the resolution will be improved already by the second failing test pattern for the analyzed circuits in the case of using 5 SAs can be seen in Figure 4.4. After the second failing pattern the improvement in diagnostic resolution slows down significantly since the accuracy of fault location is reaching its limits.

4.5.3 Trade-off between time cost and accuracy of the fault diagnosis

Table 4.4 shows how the diagnostic resolution (Res) in the case of 5 SAs can be improved by increasing the number of failed patterns to be found, i.e. by increasing the test length (Length). The data in Table 4.4 and in Figure 4.4 are averages over all the stuck-at faults in the circuits compared.

Table 4.4 *Influence of the test length on the resolution*

# Failed patterns	c2670		c5315		c7552	
	Res	Length	Res	Length	Res	Length
1	27.5	5.3	28.8	5.8	45.1	6.1
2	7.1	7.7	4.2	8.9	7.4	9.1
3	4.4	9.1	2.4	10.2	3.4	10.6
4	3.9	9.9	2.1	10.9	2.7	11.5
5	3.7	10.6	2.0	11.6	2.4	12.4
6	3.7	11.2	2.0	12.2	2.3	13.1
7	3.5	11.7	2.0	12.8	2.2	13.8
8	3.5	12.3	2.0	13.4	2.2	14.6
All	3.3	29.5	1.9	38.9	2.0	88.0

In Table 4.5 the impact of the number of SAs on the diagnostic resolution is shown. Three ISCAS circuits are compared and only the average and worse resolutions are calculated over all possible faulty cases. The best resolution for all circuits was 1. Since the influence of the multiple SAs is high especially in the beginning of diagnosis, the diagnostic procedure was stopped at the first failing pattern to determine more exactly the sensitivity of the resolution on the number of SAs. The short test sequence (diagnosis on the basis of a single failed pattern) is the reason why the resolution values (numbers of suspected faults) in Table 4.5 are rather high.

Table 4.5 *Influence of the number of SAs on the resolution*

#SA	c2670		c5315		c7552	
	Av	Worse	Av	Worse	Av	Worse
1	151.5	379	232.8	676	262.4	806
2	73.6	190	92.0	342	106.8	364
3	43.9	146	55.7	228	95.2	433
4	34.9	113	41.0	165	65.6	333
5	27.3	87	40.7	220	47.9	217
6	25.4	120	27.6	117	45.7	214
7	24.8	130	23.7	141	41.4	195
8	21.5	108	23.8	156	34.6	158
9	21.7	112	21.3	97	34.4	169
10	21.4	115	19.9	83	33.7	154

In Table 4.6 the impact of the selected number of DPs on the diagnostic resolution as well as on the test length for circuit c2670 is shown. Experimental

results are presented for the cases when were used 1, 5 and 10 SAs (column "# SA"), the diagnostic procedure was stopped after the 1, 2, 3, 10 and all failing patterns were detected, and for diagnosis were used initial set of effective test patterns selected as DPs (1*x*). The notations 2*x*, 4*x* and 8*x* mean which DPs were used for fault diagnosis from the initial set of DPs (2*x* – each second, 4*x* – each fourth, 8*x* – each eighth), i.e. how many times less DPs were used compared to the initial number of DPs. Circuit c2670 is tested by the pseudorandom sequence of length 22862 where only 116 patterns are effective, i.e. selected as initial DPs. Thus, in the case of 8*x* for diagnosis are used only 116 / 8 = 14.5 → 15 patterns that saves the BIST memory since less diagnostic data must be stored and also reduces the length of test since searching tree becomes much smaller, however, it has a negative impact on the diagnostic resolution.

Table 4.6 *Influence of the number of DPs on resolution and test length for c2670*

# SA	# DP	Diagnostic resolution					Test length				
		# failed patterns					# failed patterns				
		1	2	3	10	All	1	2	3	10	All
1	1x	151.5	24.3	8.2	4.4	4.1	5.3	8.7	10.7	16.4	37.2
	2x	295.7	64.6	29.2	5.2	4.3	4.3	7.4	9.3	15.6	28.3
	4x	519.5	204.1	114.0	46.7	39.0	3.5	5.9	7.7	14.0	21.8
	8x	831.6	473.5	346.8	201.0	201.0	2.6	4.3	5.7	11.9	14.2
5	1x	27.5	7.1	4.4	3.4	3.3	5.3	7.7	9.1	13.3	29.5
	2x	47.2	12.1	7.4	**3.7**	3.5	4.3	6.7	8.0	**12.6**	22.5
	4x	72.7	26.5	15.5	9.4	8.7	3.5	5.5	6.8	11.7	17.8
	8x	99.9	43.1	32.0	16.7	14.7	2.6	4.2	5.3	10.2	12.1
10	1x	18.2	5.9	4.2	3.4	3.3	5.3	7.5	8.7	12.8	29.0
	2x	27.8	9.0	6.0	3.6	3.5	4.3	6.4	7.6	12.1	22.0
	4x	43.0	18.4	12.1	8.3	7.8	3.5	5.4	6.6	11.4	17.4
	8x	57.4	27.6	21.4	12.1	11.4	2.6	4.1	5.2	10.0	11.9

The main message of the Table 4.4, Table 4.5 and Table 4.6 is to show how the proposed method allows to make a trade-off between the time cost (test length), the accuracy (resolution) of the fault diagnosis, and also the hardware cost (DPs stored in memory, implementation of a set of SAs). For example, in the case of the circuit c2670 and using a single SA the diagnostic resolution is 4.1 at the test length 37.2 (Table 4.6). In this case all the failing patterns will be found which means that the average diagnostic resolution 4.1 is the best possible, however the test length is rather high. When using 5 SAs (Table 4.4) even better resolution 3.7 will be achieved at the cost of test length only 10.6 which is 3.5 times better than using a single SA. The resolution can be further improved up to 3.3, however at the cost of increasing test length up to 29.5. According to Table 4.6, one of the best solutions for fault diagnosis of circuit c2670 is to exploit 5 SAs, 2*x* DPs and to stop the diagnostic procedure when 10 failed patterns are found which results the average diagnostic resolution 3.7 and the test length 12.6.

Figure 4.5 illustrates the trade-off between number of SAs, average diagnostic resolutions and test lengths for three circuits c2670, c5315 and c7552. The figure is based on data presented in Table 4.4 and Table 4.5, where additionally average resolutions and test lengths for three circuits were calculated. Figure 4.5 demonstrates that initial diagnostic procedure application time, which is proportional to the number of test sessions that was equal to 62.6, where was used a single SA and all failing patterns were targeted during fault diagnosis, could be speed up approximately by 5.4 times. To achieve it, a set of 5 SAs must be introduced and not more than 5 failing patterns must be used for fault diagnosis. This approach guaranties the average test length of 11.5 while the average diagnostic resolution is even improved from initial 3.0 to 2.7. Thus, presented tables and figures in this section give an overview of possible strategies for implementing efficient fault diagnosis for ISCAS'85 circuits.

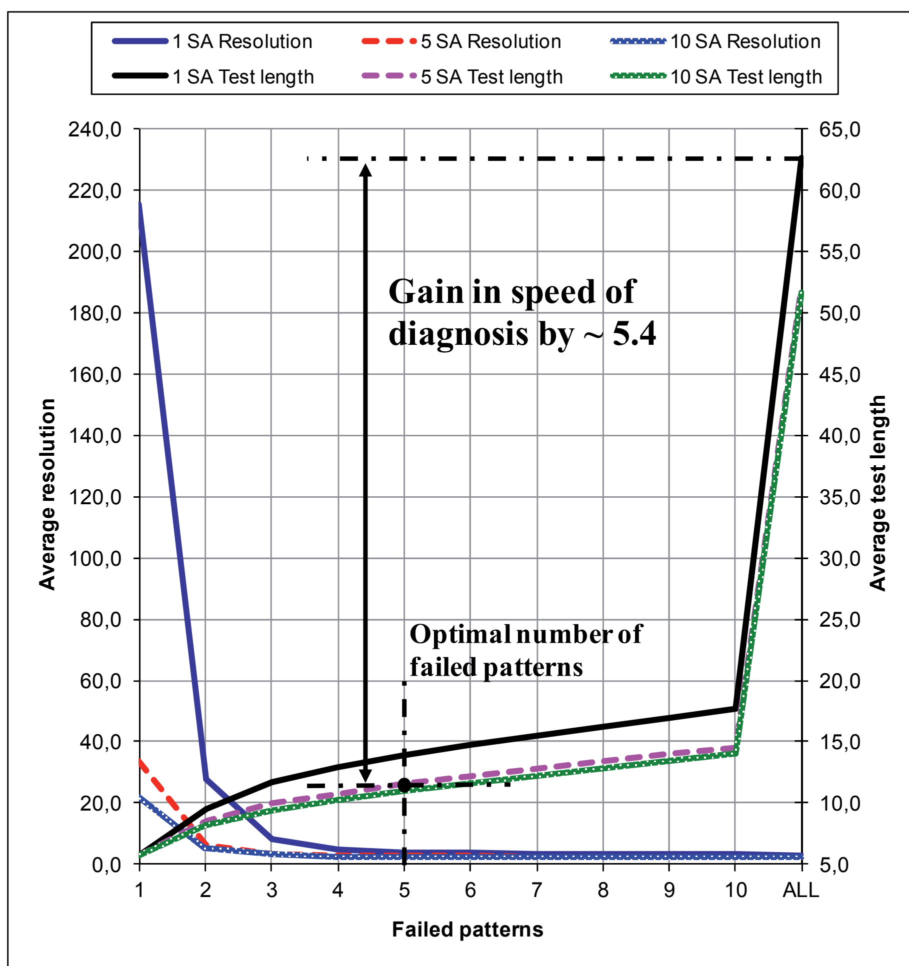


Figure 4.5 *Trade-off between resolution, test length and number of SAs*

4.6 Conclusions

A method is proposed for embedded fault diagnosis in digital systems with BIST environments:

- the proposed approach employs compressed fault tables to minimize the amount of memory space needed to store essential diagnostic data derived from a long sequence of pseudorandom test patterns
- a method for optimized partitioning of a single signature analyzer into a set of analyzers has been developed to improve the fault resolution when using the information only from small sets of failed patterns
- algorithms are proposed to design an optimal interface between the circuit under test and the set of signature analyzers to achieve the best diagnostic resolution for given test set; the partition is done only once in a single way for the whole diagnosis procedure and in this way it does not bring any additional increase in the area overhead compared to the case of a single signature analyzer
- an overview of possible strategies for fault diagnosis is presented where a trade-off analysis between the time cost (test length) and the accuracy (resolution) is carried out

Experimental results demonstrate that the proposed method for increasing the diagnostic resolution and optimizing diagnosis application time is feasible and efficient.

Chapter 5

FAULT MODEL FREE HIGH-LEVEL FAULT DIAGNOSIS

This chapter introduces a concept of fault model free diagnosis combined with cause-effect analysis in digital systems represented as networks of functional blocks. The diagnosis is considered as a task to locate a faulty block in the network by using concise block-level topological fault dictionary. The dictionary does not need fault simulation and represents only the connectivity of blocks to observable checkpoints. The distances between the entries (codewords) in the dictionary are defined, and they are used to match the observed test responses to the entries of the dictionary. A measure is proposed for evaluating the block-level diagnosability of a given network which can be used for redesign of the circuit to improve the exactness of locating the faults or faults regions in digital circuits. Experimental results provide the data which characterize the proposed measure and show the efficiency of using topological fault dictionaries for the ISCAS benchmark family.

5.1 Block-level fault-model free diagnosis

Consider a combinational circuit C with a set of observable outputs (e.g. primary outputs) OUT, a set of controllable by test inputs IN, and a set of internal fanout nodes FN. Denote by $FB \subset FN$ a subset of nodes on fanout branches from the internal nodes of FN. Consider the circuit C as a topological network of blocks (subcircuits) where each block $s_i \in S$ represents a *fanout-free region* (FFR) with an single output node $y_i \in OUT \cup FN$ and with a set of input nodes $X_i \subset IN \cup FB$. In particular case the block $s_i \in S$ may consist of a single node of a primary input if it

has fanout branches. The connections between blocks may be logic signal lines or buses. The functionalities of blocks may be represented by logical and arithmetical expressions or by procedures. The objective of the diagnosis is to locate the faulty block $s \in S$ in a set of all networked blocks of S by applying a test T and analyzing the test response $R(T)$ on the primary outputs of the circuit.

Represent the network C as a directed graph $G = (S, \Gamma)$ where the nodes represent the blocks $s \in S$ and Γ is a mapping from S to S. Here $\Gamma(s) \subset S$ denotes the set of successor nodes (blocks) of s, and $\Gamma^{-1}(s) \subset S$ denotes the predecessor nodes (blocks) of s. By $\Gamma^*(s) \subset S$ and $\Gamma^{-1*}(s) \subset S$ the transitive closures of $\Gamma(s) \subset S$ and $\Gamma^{-1}(s) \subset S$ are denoted respectively. For all the nodes $s_j \in S$ which represent the output blocks of the network, j: $y_j \in OUT$, is valid $\Gamma(s_j) = \varnothing$. For all the nodes $s_j \in S$ which represent the blocks with only primary inputs of the network, j: $X_j \subset \mathrm{IN}$, is valid $\Gamma^{-1}(s_j) = \varnothing$.

An example of such a graph is presented in Figure 5.1, where each node represents a FFR of the circuit. The task of the fault diagnosis, instead of locating predefined faults (e.g. stuck-at faults) inside FFRs, will be to locate the faulty FFR if the circuit has failed during test. There will be no restrictions to the fault types inside the FFRs. The only restriction for sake of the ease of calculating the measure of diagnosability will be the assumption of a single faulty block. However, the measure can be used also for estimating the diagnosability of a system in a general case of multiple faulty blocks.

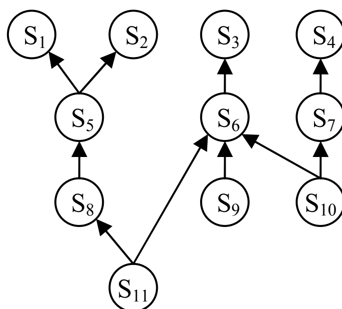


Figure 5.1 *A graph representing a network of blocks*

5.2 Diagnostic matrix of a network

The block-level topological fault dictionary for the given combinational circuit C is represented as a *diagnostic matrix* $DM = \| d_{ij} \|$ where i denotes a node $s_i \in S$ (a block in the network) and j denotes an observable output node $s_j \in S$ where $\Gamma(s_j) = \varnothing$. The notation $d_{ij} = 1$ is used, if there is a path from s_i throughout the

graph G to the output node s_j, otherwise $d_{ij} = 0$. The notation $d_{ij} = 1$ means that the erroneous value on the output of the block $s_i \in S$ can be propagated to the observable primary output $y_j \in OUT$ of the network. The subset of all observable nodes reachable from the output of the block s_i can be calculated as $OUT(s_i) = \Gamma^*(s_i)$.

Let call the row vectors $CW_i = (d_{i1}, d_{i2}, \dots, d_{in})$ of the matrix DM as *diagnostic codewords*. Here $n = |OUT|$ is the number of observable nodes (primary outputs) of the network. Each block s_i has its own binary codeword where $d_{ij} = 1$ means that a fault in the block s_i is able to change the value of the observable output y_j if a signal path from the output of the block s_i up to $y_j \in OUT$ is activated. Several blocks may have identical diagnostic codewords.

The procedure for generating the diagnostic matrix from the graph G which describes the structure of the circuit is as follows.

Algorithm 5.1:

For all j: $\Gamma(s_j) = \varnothing$
BEGIN
$d_{jj} = 1$
For all k: $d_{kj} \in \Gamma^{-1}*(s_j)$ set $d_{kj} = 1$
END

An example of the diagnostic matrix DM created from the graph G in Figure 5.1 according to the *Algorithm 5.1* is shown in Table 5.1.

Table 5.1 *Diagnostic matrix of the circuit in Figure 5.1*

	y_1	y_2	y_3	y_4
s_1	1			
s_2		1		
s_3			1	
s_4				1
s_5	1	1		
s_6			1	
s_7				1
s_8	1	1		
s_9			1	
s_{10}			1	1
s_{11}	1	1	1	

In this approach is assumed that all the outputs of the network can be observed independently, in other words, it is assumed that each primary output has its own *signature analyzer* (SA). Because of the redundancy of pseudorandom test sequences in the sense of repeated detection of faults, it may be expected that if there is a fault in a block s_i of the network, this fault will show it as an erroneous

output signal at least once on each primary output of the network, corresponding to the *diagnostic codeword* CW_i of the block.

In the case of embedded diagnosis when not all primary outputs are separately observable, and the groups of outputs are merged to joint SAs the diagnostic matrix should be respectively modified. To represent the diagnostic model of such a case of a multi-SA structure (see Chapter 4) where the number of SAs is less than the number of primary outputs of the network, a new diagnostic matrix $DM^m = ||d^m_{ik}||$ must be constructed, where $k = 1, 2, ..., m$, and m is the number of SAs. The matrix DM^m will be generated from DM by merging the columns for the outputs connected to the same SA. Suppose that the subset of outputs $OUT_k \subset OUT$ is connected to the signature analyzer SA_k. Then the entries of the column k in DM^m are calculated as logic sums:

$$\forall i\colon d^m{}_{ik} = \bigvee_{j \in OUT_k} d_{ij}$$

where OUT_k is the subset of columns in DM to be merged in DM^m.

The diagnostic matrix DM can be used for fault diagnosis in a similar way as traditional fault tables or fault dictionaries by looking for matches between the diagnostic results and the codewords in DM. However, without special manipulations to be explained later the diagnosis may remain inaccurate or even impossible.

Denote by $R(T) = (r_1, r_2, \ldots, r_n)$ the test response where $r_j = 1$ when a fault has caused an error during the test on the output node $y_j \in OUT$ at least once, and $r_j = 0$ otherwise. Introduce the following *diagnostic property* of the test which is used for fault diagnosis.

Property 5.1. All the possible faults in the block s_i in the network C are activated during the test at least once to all observable nodes $OUT(s_i)$ reachable from s_i.

Assume first, that the *Property 5.1* is valid. In this ideal case, the faults in a block s_i will always produce a match $R(T) = CW_i$ in the fault dictionary. *Property 5.1* may be valid also partially, for a subset of all faults.

In general case, it may happen that $R(T) \neq CW_i$. However, it is not difficult to realize that in general case always $R(T) \leq CW_i$, i.e. $\forall j, j = 1, 2, \ldots, n\colon r_j \leq d_j$. This statement follows from the assumption of a single faulty block in the network.

As an example, consider the network in Figure 5.1, and the diagnostic matrix in Table 5.1. The response for a test experiment T will be represented as a 4-bit codeword $R(T)$. The rows of the matrix DM will correspond to the expected codewords for different faulty cases as responses to the test experiment (in the case when the *Property 5.1* of the test is valid). For example, if $R(T) = 0011$, then from DM can be concluded that the block s_{10} should be faulty. On the other hand, in the

case of the codeword 0010 the diagnosis will be ambiguous, since there are three rows in the table with such codeword. The subset of suspected faulty blocks in this case is: $\{s_3, s_6, s_9\}$.

The general case, when the *Property 5.1* is not valid, will be considered in section 5.4.

5.3 Lower bound of average block-level diagnosability

Consider the ideal case when according to *Property 5.1* all the possible faults in a block s_i will be propagated during the test to all nodes of $OUT(s_i)$. This property may be used as a criterion to be considered during test generation for block-level diagnostics purposes. For example, in the case of embedded diagnosis based on pseudorandom sequences produced by BIST, the probability of propagating faults to all reachable outputs will be the higher the longer is the test sequence. The test generation problem itself is not discussed here.

Lemma 5.1. If the two codewords CW_k and CW_m in the matrix *DM* are different $CW_k \neq CW_m$, then the faults in the corresponding blocks s_k and s_m are distinguishable.

Proof. Since $CW_k \neq CW_m$, there exists at least one column j in *DM* where $d_{kj} \neq d_{mj}$ which means that only one of these two blocks is able to influence the value of $y_j \in OUT$. Consequently, in the case when a faulty behaviour is detected with either $R(T) = CW_k$ or $R(T) = CW_m$, then the faulty or non-faulty behaviour of $y_j \in OUT$ will exactly explain which of these two blocks is faulty and which is not.

From the *Lemma* the following *Corollary* results.

Corollary. If the two codewords CW_k and CW_m in the matrix *DM* are equal then the faults in the corresponding blocks s_k and s_m are indistinguishable with the given method.

From the *Corollary* and the *Property 5.1* it follows that in order to improve the diagnostic resolution it is needed either to improve the test (when *Property 5.1* is not valid), or to redesign the circuit e.g. by adding observable checkpoints in such a way that in the new diagnostic matrix the codewords CW_k and CW_m become different.

Theorem 5.1. If all the codewords in the matrix *DM* are different from each other, then all the faults in the network *C* are distinguishable with accuracy of block locations.

The proof of the *Theorem 5.1* follows from the *Lemma 5.1*.

Let $| S |$ be the number of rows in *DM* of a network *C* where *S* is the set of all blocks to be determined as faulty or not faulty. Partition all the blocks in *S* into a set of groups $M = \{M_k\}$, so that the codewords of the blocks in a particular group

$M_k \in M$ are the same. Let $|M_k|$ be the number of blocks with the same codeword CW_k in M_k. Obviously, $|M| \leq |S|$, and $|M| = |S|$ only in the case when all the rows in *DM* are different.

Taking into account the indistinguishability of faults in $M_k \in M$, redefine now the task of diagnosis.

The objective of the diagnosis is to locate the faulty group of blocks $M_k \in M$ by reasoning the test response *R(T)*.

Now the average block-level diagnosability of the given network *C* can be calculated as follows:

$$D = \frac{\sum_{k=1}^{|M|} |M_k|}{|M|} \tag{5.1}$$

As an example, for the diagnostic matrix *DM* in Table 5.1 the following partition is obtained:

$$M = \{\{s_1\}, \{s_2\}, \{s_3, s_6, s_9\}, \{s_4, s_7\}, \{s_5, s_8\}, \{s_{10}\}, \{s_{11}\}\}$$

In this partition, there are three groups of indistinguishable faulty blocks: $M_3 = \{s_3, s_6, s_9\}$, $M_4 = \{s_4, s_7\}$, and $M_5 = \{s_5, s_8\}$. The average diagnosability of the network according to (5.1) is $D = 1.57$.

The value of *D* refers to the block-level diagnostic resolution achievable by the diagnostic test with *Property 5.1*. In other words, the value of *D* means the average number of suspected blocks as the result of fault diagnosis.

For the general case when the *Property 5.1* is not valid, two possibilities may take place during diagnosis: (1) no match will be found for *R(T)* in *DM*, or (2) the match will be „wrong“, in other words, the resolution of the diagnosis will be worse. The both cases will be discussed in the next section.

As can be seen further, in the general case when *Property 5.1* is not valid, the block-level diagnostic resolution may be worse, and the number of suspected blocks may be higher than *D* calculated by (5.1). Therefore, the value of the formula (5.1) can be interpreted as the lower bound of average block-level diagnosability.

5.4 Diagnosis with adjusting the test responses

Consider now the general case where the *Property 5.1* is not valid for the given test, i.e. the case where a fault located in the circuit during the test sequence will influence not all the primary outputs dictated by the diagnostic codeword of the block containing the fault.

Denote by $E(CW_i)$ the number of ones in the codeword CW_i. Then the codeword CW_i is called *reducible* to CW_j if $E(CW_i) > E(CW_j)$ and CW_j can be produced from CW_i by only changing ones to zeroes. The codewords are called *comparable* if one of them is reducible to another. The notation $CW_i > CW_j$ is used when CW_i is *reducible* to CW_j.

Denote by $L(i,j)$ the *Hamming distance* between two *comparable* codes CW_j and CW_i calculated as $L(i,j) = E(CW_i) - E(CW_j)$. Introduce a *codeword distance graph* $\Omega = (CW, \Delta)$ where CW is the set of all codewords in DM, and Δ represents a mapping from CW to CW. Denote by $\Delta(CW_i) \subset CW$ the subset of the *closest smaller codewords* for $CW_i \in CW$. The nodes $CW_j \in \Delta(CW_i)$, $CW_i > CW_j$, are successors of CW_i, and the nodes $CW_j \in \Delta^{-1}(CW_i)$, $CW_i < CW_j$, are predecessors of CW_i in Ω. Denote by $\Delta^*(CW_i)$ and $\Delta^{-1*}(CW_i)$ the transitive closures of $\Delta(CW_i)$ and $\Delta^{-1}(CW_i)$, respectively. Note that there exists a connection between two nodes CW_i and CW_j in Ω only if the codewords CW_i and CW_j are *comparable*.

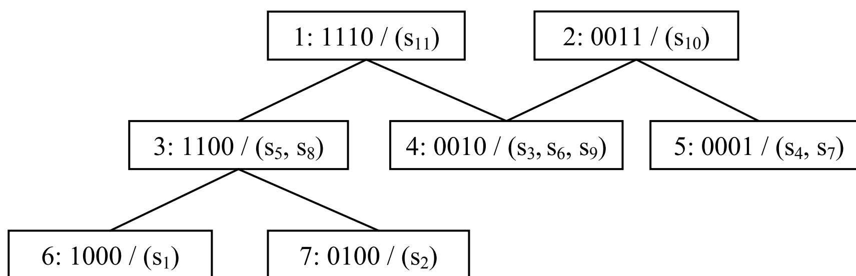


Figure 5.2 *The codeword graph of the network in Figure 5.1*

An example of the codeword distance graph Ω for the network in Figure 5.1 is represented in Figure 5.2. The notation in the nodes is: i: CW_i / M_i.

During the fault diagnosis with a test T in case of a faulty block $s \in M_i$, the following cases may happen:

(1) $R(T) = CW_i$

(2) $R(T)$ does not match with any of the codewords in Ω

(3) $R(T)$ matches with another codeword CW_j which does not correspond to M_i

(1) The first case corresponds to the correct diagnosis, the other cases mean a *distortion* of the diagnosis because of not detecting the fault on all outputs $OUT(M_i)$ dictated by the codeword due to the fault no propagated to all outputs it may affect, or due to aliasing taking place in SAs.

(2) Suppose that $R(T) \neq CW_i$, and there is no match between $R(T)$ and any other CW_j in Ω. Assume, also that there is no CW_j in Ω, such that $R \leq CW_j < CW_i$. Then,

$R(T)$ can be corrected to match with CW_i , and produce the correct diagnosis according to the following theorem.

Theorem 5.2. If there exists no codeword CW_k , so that $CW_j < CW_k < CW_i$. Then all test responses $R(T)$, where $CW_j < R(T) < CW_i$, can be corrected to CW_i referring exactly to the suspected set of faulty blocks M_i .

Proof. Suppose by a contradiction that there is another faulty set of blocks M_k , $k \neq j$, able to produce the same test response $R(T)$. Then two cases are possible, either $CW_k = CW_i$, if *Property 5.1* is valid for the given fault, or $R(T) = CW_k < CW_i$ which both are in contradiction with the presumption of the *Theorem 5.2*.

(3) Assume now that $E(CW_i) - E(R(T)) \geq k$, and there is a node CW_j in Ω, so that $R(T) = CW_j$. In this case the faulty block may locate in a subset of blocks represented by a subset of codewords $\Delta^*(CW_i) \cap \Delta^{-1*}(CW_j)$. This means a decrease of the diagnostic resolution.

As an example, consider in Figure 5.2 the case when there is a fault in s_{11} and $R(T)$ = 0110. There is no such codeword in the graph. However, $R(T)$ can be adjusted to 1110, and the correct diagnosis s_{11} will be achieved. As a second example, let $R(T)$ = 1100. There is a match now with $\{s_5, s_8\}$. If can be proven that s_5 and s_8 are correct, the diagnostic procedure should continue in the blocks represented by $\Delta^*(CW_i) \cap \Delta^{-1*}(CW_j) = \Delta^*(1110, 1100, 0010) \cap \Delta^{-1*}(1100, 1110)=$ = (1110), i.e. in the block s_{11} . Because of the test with less diagnostic capability the diagnostic resolution was also less – instead of only s_{11} , three blocks $\{s_5, s_8, s_{11}\}$ are suspected as fault candidates.

5.5 Probabilistic diagnosability measure

Let return to the case (3) in the previous section, where because of *the distortion* $R(T)$ matches with another codeword CW_j in Ω which does not correspond to M_i . Then following assumptions are valid, $E(CW_i) - E(R(T)) \geq k$ and $R(T) = CW_j$. Regarding the diagnosability, this means that the suspected candidate set of faulty blocks should be increased:

$$M_i^* = \Delta^*(CW_i) \cap \Delta^{-1*}(CW_j) \tag{5.2}$$

In such a way, by using formula (5.2) to each possible faulty situation, a set of blocks can be determined as candidates for being faulty at a given response $R(T)$. From the discussion above, a measure follows for a probabilistic diagnosability:

$$D = \frac{1}{n}\sum_{i=1}^{n}(p_i \cdot |M_i^*|) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|F_i|}{|F|} \cdot |M_i^*|\right) \tag{5.3}$$

where n is the number of nodes in the codeword graph Ω, M_i^* is the set of blocks suspected as candidates for being faulty with probability p_i . For a particular case, when the possible faults can be enumerated, and the faults have equal probabilities,

the probabilities in (5.3) can be calculated as $p_i = |F_i| / |F|$, where F is the set of all faults, and F_i is the set of faults in M_i.

In case a set of faults is enumerable the following algorithm can be used for calculating the probabilistic diagnosability.

Algorithm 5.1:

1. Calculate the reference R for the given test sequence T
2. For all nodes CW_i of the graph Ω
3. Set up *Cluster* (CW_i) of blocks consisting of a node CW_i
4. For all blocks M_i of the node CW_i
5. For all faults f in the block M_i
6. Calculate output response $R(f)$
7. Find the closest codeword CW_j to $R(f)$
8. Update $Cluster\ (CW_i) = Cluster\ (CW_i) \cup (\Delta^*(CW_i) \cap \Delta^{-1*}(CW_j))$

   END_for_all_faults
9. Calculate the number of blocks in *Cluster* (CW_i): $M_i^* = |Cluster\ (CW_i)|$
10. Calculate the number of faults F_i in block M_i
11. Calculate the probability $p_i = F_i / F$

    END_for_all_blocks

    END_for_all_nodes
12. Calculate the average diagnosability according to (5.3)

    END_of_Algorithm

5.6 Improving the diagnosability

To improve the diagnostic resolution, the original network can be redesigned for better diagnosability by inserting additional observable checkpoints in such way that in the final diagnostic matrix all the rows will be different.

If an error is detected by the test response $R(T)$, and a match $R(T) = CW_i$ is found, then a fault is expected to be in the subnetwork represented by the subset of blocks $M_i \in M$. If $|M_i|$ is too big, the diagnostic resolution may not be satisfactory.

Consider a network C as a graph $G = (S, \Gamma)$, and the subnetworks of blocks M_i with same codewords CW_i as subgraphs $G_i = (M_i , \Gamma)$ where $M_i \subseteq S$. Denote $n_i = |M_i|$, $N = |S|$, and $r = |M|$ is the number of subsets of blocks with the same codeword.

All the blocks $s \in M_i$ in any M_i may be the candidates to become a checkpoint to improve the diagnostic resolution (diagnosability) of the system. By choosing $s \in M_i$ as a checkpoint then a partitioning of $M_i = M_{i,1} \cup M_{i,2}$ is obtained so that $M_{i,1} \cap M_{i,2} = \varnothing$. Denote $M_{i,1} = \Gamma^* (s) \cap M_i$ and $M_{i,2} = \Gamma^{-1*} (s) \cap M_i$, where

$s \in M_{i,2}$. Because of adding a new checkpoint (observable node) the length of all codewords will increase by 1, and the subsets of blocks $M_{i,1}$ and $M_{i,2}$ will have now different codewords. As the result, the faulty blocks in $M_{i,1}$ and $M_{i,2}$ become distinguishable. An example of the impact of adding a checkpoint to the node s_6 in Figure 5.1 is shown in Table 5.2.

Table 5.2 *Diagnostic matrix after adding a checkpoint*

s_i	Codeword bits					Partitions	
	1	2	3	4	5	M_i	$M_{3,e}$
s_1	1	0	0	0		M_1	
s_2	0	1	0	0		M_2	
s_3	0	0	1	0	0	M_3	$M_{3,1}$
s_6	0	0	1	0	1		$M_{3,2}$
s_9	0	0	1	0	1		
s_4	0	0	0	0		M_4	
s_7	0	0	0	0			
s_5	1	1	0	0		M_5	
s_8	1	1	0	0			
s_{10}	0	0	1	1		M_6	
s_{11}	1	1	1	0		M_7	

Assume the probabilities that a block is faulty are equal for all blocks in the network. Then, $p_i = n_i / N$, is the conditional probability that in case of detecting a fault the cause of the fault is in M_i . Denote the number of blocks in the partitioned subnetwork $M_{i,2}$ with checkpoint by $m_i = |\Gamma^{-1}*(s) \cap M_i|$.

To select sequentially depending on the test responses the optimized set of checkpoints to achieve as maximum improvement of diagnosability as possible by minimum inserted checkpoints the following *Theorem* can be used.

Theorem 5.3. To achieve the maximum improvement in diagnosability of the network $G = (S, \Gamma)$ with subnetworks $G_i = (M_i, \Gamma)$, $i = 1, 2, ..., r$, where $M_i \subseteq S$, the checkpoint should be inserted to the block $s \in M_i$ where $m_i(n_i - m_i)$ is maximum.

Proof. Consider the initial diagnosability of a system as:

$$D = p_1 n_1 + p_2 n_2 + ... + p_i n_i + ... + p_r n_r \qquad (5.4)$$

After insertion of the checkpoint to a block $s \in M_i$ the diagnosability will be changed:

$$D' = p_1 n_1 + p_2 n_2 + ... + p_i [p(s) m_i + (1 - p(s))(n_i - m_i)] + ... + p_r n_r \qquad (5.5)$$

where $p(s) = m_i / n_i$ is the conditional probability that in case of faulty subnetwork M_i the fault will be located in $M_{i,2} \in M_i$.

To have $D - D'$ = max, the following must be valid:

$$p_i n_i - p_i \left[p(s)m_i + (1 - p(s))(n_i - m_i) \right] = \max \quad (5.6)$$

After substituting the probabilities and dropping for simplicity the subscripts:

$$\frac{n^2}{N} - \frac{n}{N}\left(\frac{m^2+(n-m)^2}{n}\right) = \frac{n^2-m^2-(n-m)^2}{N} = \frac{2m(n-m)}{N} = max \quad (5.7)$$

From that it follows that to achieve the maximum improvement in diagnosability at each step the checkpoint should be inserted to the node $s \in M_i$ where $m_i(n_i - m_i)$ is maximum.

As an example, consider the network in Figure 5.1 which according to (5.4) has D = 1.9. The value of $m_i(n_i - m_i)$ will be 2 (maximum) for the blocks s_6 and s_9, 1 for the blocks s_7 and s_8, and 0 for other blocks. If choosing either s_6 or s_9 for the checkpoint then D' = 1.54 which is the maximum increase in diagnosability by a single additional checkpoint. If choosing either s_7 or s_8 a worse result will be obtained D' = 1.73.

5.7 Experimental data

The goal of the experimental research was to evaluate the conception of fault model free diagnosability by using the ISCAS'85 [92] [93] and ISCAS'89 [95] benchmark circuits. The reason of choosing these circuits was twofold: (1) since the objectives of this research are the networks of components where only the topology of connections is of importance and not the functionality of components, these benchmark circuits provide rather complex topologies; (2) second, the circuits allowed to demonstrate the feasibility of using the proposed approach also at the logic level. In the experiments, the *fan-out free regions* (FFRs) of the circuits were considered as blocks to be diagnosed as faulty or fault-free.

In Table 5.3 the following data are shown: *#Out* – number of outputs, *#Blocks* – number of blocks, *#Groups* – number of groups with the same codeword, *D_B* – average block-level diagnosability calculated according to formula (5.1), i.e. the lower bound of average block-level diagnosability, *D_P* – average block-level diagnosability calculated according to formula (5.3), i.e. the probabilistic diagnosability with assumption that all blocks have equal probabilities to fail and *Property 5.1* is valid, *Max* – maximal number of suspected blocks with the same codeword (the minimal number for all circuits was 1), and *HD* – average Hamming distance between the codewords. The lower bound of average block-level diagnosability *D_B* of all the circuits except c6288 is 2.6 which indicates a rather high inherent diagnosability of ISCAS benchmark circuits. The circuit c6288 representing a 32-bit multiplier is an exceptional case, where despite the huge number of possible codewords (2^{32} = 4294967296) there is only 63 different codewords (groups) compared to a large number of blocks 1488. Thus, the most blocks influence on the same outputs of the circuit (a large number of blocks with

the same codeword), causing in this way very low block-level diagnostic resolution. Column *Max* and *D_P* (*probabilistic diagnosability*) indicate the circuits which contain large groups of indistinguishable blocks and where the achieving high block-level diagnostic resolution during fault diagnosis will be complicated without redesigning the circuit for better diagnosability. For instance, in circuit c1355 the largest group consists of 185 indistinguishable blocks whereas there are 291 blocks in all. If all blocks have equal probabilities to fail then in $(185 / 291) \times 100\% = 63.5\%$ of cases the block-level diagnostic resolution will be equal to 185. However, practically all blocks have different sizes, i.e. contain different number of gates and, consequently, different number of possible defects. Thus, all blocks do not have equal probabilities to fail and the real probabilistic diagnosability can be considerably better (or even worse). On the other hand, the ISCAS'89 circuits demonstrate very high inherent diagnosability, where the worst *probabilistic diagnosability* $D_P = 7.3$ is obtained for circuit s5378 and the best $D_P = 1.7$ is for circuit s832.

Table 5.3 *Block-level diagnosability*

Circ.	#Out	#Blocks	#Groups	D_B	D_P	Max	HD
c499	32	187	43	4.3	69.0	113	15.1
c880	26	151	68	2.2	5.2	15	3.0
c1355	32	291	43	6.8	118.6	185	15.4
c1908	25	248	48	5.2	69.1	129	7.6
c2670	140	430	206	2.1	12.3	41	2.3
c3540	22	378	111	3.4	11.5	33	2.4
c5315	123	633	350	1.8	14.9	64	5.2
c6288	32	1488	63	23.6	60.9	91	8.3
c7552	108	920	315	2.9	20.2	48	4.2
s832	24	63	50	1.3	1.7	4	4.6
s1196	32	187	138	1.4	2.1	8	3.2
s1423	79	259	181	1.4	2.5	12	11.2
s3271	130	555	377	1.5	5.4	42	3.3
s5378	228	1083	662	1.6	7.3	28	4.2
s15850	684	2202	1477	1.5	4.8	42	16.4
s13207	790	2014	1424	1.4	4.2	60	5.1
			Average	3.9	25.6	57.2	7.0

Investigations were carried out to evaluate the efficiency of correcting test responses using the Hamming distances between diagnostic codewords of network blocks. The results are depicted in Table 5.4, where as the test were used sequences of pseudorandom patterns with length in column 2 and stuck-at fault coverage in column 3. If the fault coverage did not increase any more, the sequence was interrupted. The pseudorandom test was chosen for two reasons: to get easily a test with a feature of *N*-detection [8] to make the *Property 5.1* valid for as many as possible faults and to investigate the feasibility of the approach for BIST based

embedded diagnosis. The test response for each *stuck-at fault* (SAF) has been calculated and compared with the codeword of the block where the fault was located. Here was assumed that each primary output is independently observable, i.e. it has its own SA. The columns 4 and 5 show the percentages of match and mismatch of test responses *R(T)* with correct entries CW_i in fault dictionaries (diagnostic matrices *DM*) before correction, and the columns 6 and 7, respectively, after correction.

Table 5.4 *Test response correction data for the case of stuck-at faults*

Circuit	Test Properties		Response before correction, %		Corrected response, %	
	#Patterns	FC %	Correct	Wrong	Correct	Wrong
c499	1373	99.33	51.91	47.42	99.33	0.00
c880	2694	100.00	91.85	8.15	98.79	1.21
c1355	1438	99.51	54.20	45.30	99.51	0.00
c1908	4420	99.48	51.67	47.81	98.73	0.75
c2670	22682	88.65	68.28	20.37	85.38	3.27
c3540	9631	95.54	57.43	38.11	79.67	15.87
c5315	1793	98.89	79.09	19.80	89.86	9.03
c6288	42	99.34	6.66	92.68	97.82	1.52
c7552	24337	94.09	48.80	45.28	69.01	25.08
s832	11974	98.53	98.53	0.00	98.53	0.00
s1196	29248	99.70	71.06	28.64	92.58	7.12
s1423	29848	98.76	69.26	29.50	96.61	2.15
s3271	28577	100.00	89.14	10.86	95.30	4.70
s5378	47270	98.95	76.11	22.83	98.02	0.93
s15850	78460	92.19	69.05	23.14	90.06	2.13
s13207	71017	98.07	69.63	28.44	91.15	6.92
	Average	97.56	65.79	31.77	92.52	5.04

For example, for c1908 first only 51.67% correct matches were obtained. After correcting the test response *R(T)* using *Hamming distance*, the percentage of match was raised to 98.73% whereas only 0.75% of all faults were matched to a "wrong" codeword. However, the term "wrong" means here not a wrong diagnosis, rather that only the diagnostic resolution for these faults will be worse than the lower bound given in the column *D_B* in Table 5.3. The percentages in the column 7 give the idea how far the current diagnosability remains from the lower bound of diagnosability. The "wrong" matches which reduce the diagnosability are explained by the *hard-to-test faults* (HTTF) for which the *Property 5.1* is not valid. To reduce the percentage of HTTF, the test quality should be improved to meet the *Property 5.1*.

From Table 5.3 and Table 5.4 can be seen that in general, the bigger is the average *Hamming distance* between the codewords, the more exact can be the diagnosis (the less is the number of mismatches). Rather big average *Hamming*

distance between the codewords shows that there is rather high potential to correct the „distorted" test responses to match the correct block codewords, and to reach in such a way high diagnostic resolution.

In Figure 5.3, the curves for ISCAS'85 and ISCAS'89 circuits show for every value of x on the horizontal axis the probability $P(x)$ that the *Hamming distance HD* between the real fault codewords and faulty block codewords is less or equal to x. The curve AVERAGE shows the average probabilities for the whole family of circuits. The relationships in Figure 5.3 illustrate only rough general trends of the discrepancies between the real fault codewords and faulty block codewords, and do not represent the real probabilities of diagnosis inaccuracy. The reason is that a majority of fault codewords can be rectified due to the unidirectional distortions as shown in Table 5.4. For example, in Figure 5.3 for circuit c6288 only approximately 40% of test responses that mismatch the expected codewords are distorted by 10 or less bits. However, according to Table 5.4 the procedure of adjusting test responses can reduce considerably the number of "wrong" codewords from 92.68% to 1.52% whereas the average HD is 8.3 and the number of outputs equals to 32 as shown in Table 5.3.

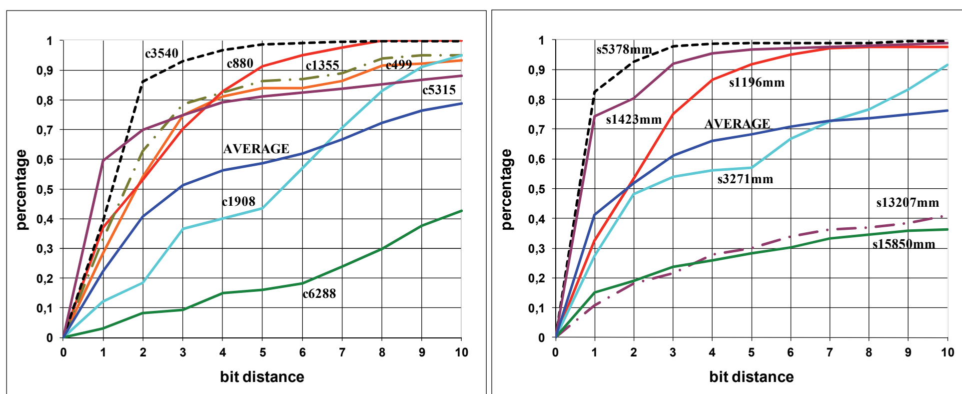


Figure 5.3 *Integrated percentages of distances between fault responses and faulty block codewords for ISCAS'85 and ISCAS'89 benchmarks*

5.8 Conclusions

In this chapter a conception was discussed to carry out block-level fault diagnosis in digital networks without using fault models. Instead of traditional SAF model based diagnosis, a method is proposed for locating faulty areas in the circuit with accuracy of FFR blocks as network components. Since no fault model is used, the method is applicable for both, debugging design errors and locating the manufacturing defects. Any type of defect, single or multiple, is allowed to take place in a block. The size of the block-level fault dictionary depends linearly on the

number of blocks to be determined as faulty or not faulty. The size of the blocks, however, will be the trade-off between the complexity of the fault dictionary and the diagnostic resolution.

A measure for characterizing the lower bound of block-level diagnosability of a given network was proposed, and was shown how this measure can be used for improving the diagnosability of networks by inserting additional observable checkpoints.

A measure of *Hamming distance* between the codewords of entries in the fault dictionary was introduced, and a fault diagnosis method was presented which uses these distances to adjust the not matching test response to match the correct entry in the dictionary to locate the faulty block. This adjustment of test responses is similar to using *Hamming distances* for correcting errors in signal transmission.

By experiments the feasibility of the block oriented fault diagnosis for a subclass of SAF model was analyzed. The results were promising: the lower bound of average block-level diagnosability was 3.9, rather high average *Hamming distance* 7.0 between codewords allowed quite efficiently to correct "wrong" test response codewords to match the available codewords in the fault dictionary.

Chapter 6
HIERARCHICAL FAULT DIAGNOSIS

This chapter introduces a hierarchical approach for fault diagnosis in combinational digital circuits represented as a network of modules. As modules either library components (e.g. complex gates) of digital circuits or arbitrary subcircuits are considered. The higher level fault diagnosis is carried out in two phases. In the first phase, faulty modules are located by cause-effect analysis using high-level fault dictionary. The size of the dictionary depends linearly on the number of modules in the circuit. In the second phase, the set of suspected faulty modules is pruned by reasoning of the defective behavior. At the lower level, the physical defects are directly located in suspected faulty modules using defect libraries of the modules or by effect-cause reasoning inside the module. The proposed approach to fault diagnosis helps to cope with the growing complexities of digital circuits. The experimental results show high module-level diagnostic resolution of the proposed approach.

6.1 General description of the method

The main objective of the proposed method is to combine the cause-effect fault diagnosis in combinational circuits with the physical defect oriented approach based on the general conditional fault model [65] [66] [67] [68] [69] [70]. The approach is hierarchical and is carried out at two levels (Figure 6.1).

At the high-level, the model of a circuit is presented as a network of modules. Each module represents a *fanout-free region* (FFR) of the circuit with a different number of gates. The modules may be library components (complex gates) and/or arbitrary single output logic subcircuits. Thus, instead of enumerating all the possible fault cases (or defects), in the high-level fault dictionary are listed only the

modules as potential faulty items in the circuit. This helps to cope with the complexity of the fault dictionaries. Initial modules can be grouped even into larger modules to reduce the size of fault dictionary. In general, there are no restrictions set to the fault types inside the modules. The task of the high-level diagnosis is to locate the candidate faulty module or a subset of candidate faulty modules.

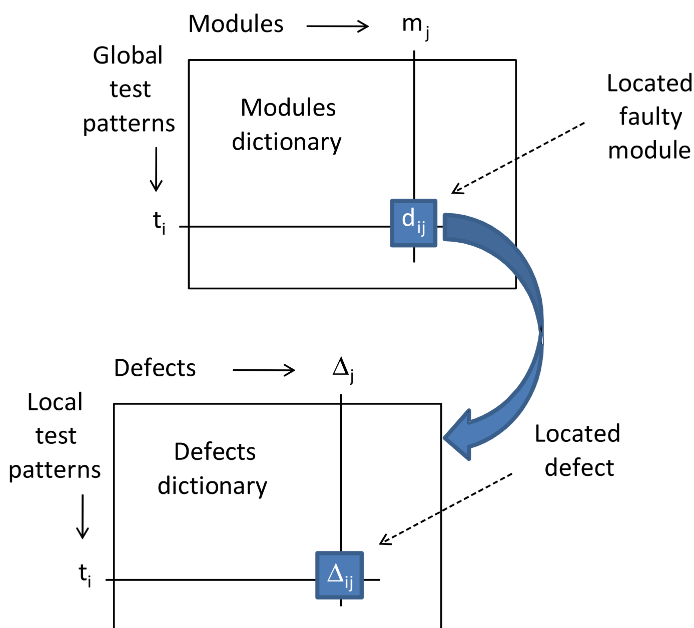


Figure 6.1 *Illustration of the hierarchical approach to fault diagnosis*

First, by cause-effect diagnosis approach based on using the high-level fault dictionary, an initial set of candidate faulty modules is determined. On the next step, by using effect-cause high-level module reasoning, the initial subset of candidate faulty modules will be suppressed (Figure 6.2). The high-level module reasoning is defect oriented; however, the direct defect analysis is avoided. The defects are modeled indirectly by using the functional fault model which is represented by pairs of *stuck-at faults* (SAFs) and defect conditions [67] [70]. In the process of reasoning, only the defect conditions are considered, which allow to avoid dealing directly with the whole huge list of defects in the circuit.

At the low-level, the fault diagnosis is carried out in the modules determined during the high-level reasoning as faulty. For low-level diagnosis purposes, the modules are represented by the sets of local test patterns which are treated as conditions for activating physical defects in modules (Figure 6.1). This diagnostic information is captured in the form of defect dictionaries and is stored in the component libraries. Using pre-generated defect dictionaries, low-level cause-effect reasoning is carried out to locate the defects in the modules. If the defect

location with the help of defect dictionaries is not possible, or if no dictionaries for given modules are available, effect-cause low-level defect reasoning should be carried out to locate the defects in suspected modules (Figure 6.2).

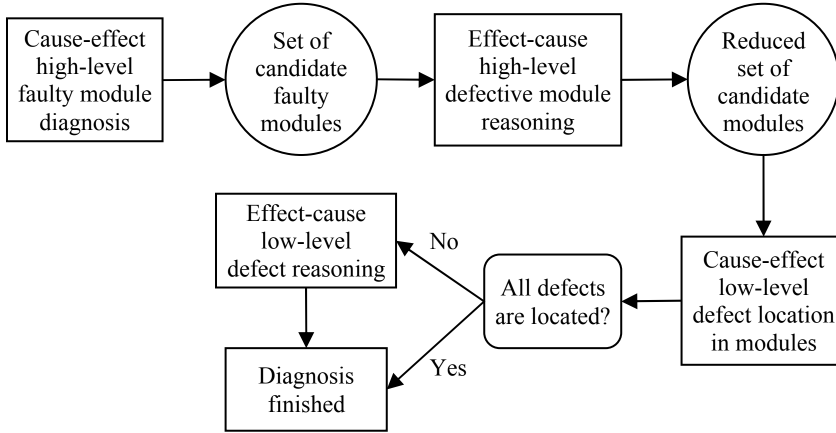


Figure 6.2 *The hierarchical fault diagnosis flow*

6.2 Cause-effect high-level fault diagnosis

Consider a combinational circuit C synthesized as a network of modules with a single output (library complex gates or arbitrary subcircuits) with a set of primary outputs OUT, a set of primary inputs IN, and a set of internal nodes INT. The network C consists of a set of modules M where each module $m_j \in M$ has an output node $y_j \in OUT \cup INT$ and a set of input nodes $X_j \in IN \cup INT$.

The circuit represents a graph $C = (M, \Gamma)$, where the nodes correspond to the modules $m \in M$, and Γ is a relation on M, where $\Gamma(m) \subset M$ denotes the successor nodes of m and $\Gamma^{-1}(m) \subset M$ is the set of the predecessor nodes of m. For all the output nodes $m_j \in M$, j: $y_j \in OUT$, is valid $\Gamma(m_j) = \varnothing$. For all the input nodes $m_j \in M$ which represent the modules with only primary inputs j: $X_j \subset IN$ is valid $\Gamma^{-1}(m_j) = \varnothing$.

Example 1. An example of a combinational network C is presented in Figure 6.3 where to each network component (functional block F) a module corresponds. The topological graph $C = (M, \Gamma)$ of the network is depicted in Figure 6.4, where are shown modules that correspond to respective functional blocks (e.g. $F_1 \rightarrow 1$) and the connections between the modules (e.g. F_1 is connected to F_4 and F_7). The task of the high-level fault diagnosis is to locate the faulty module if the testing of the circuit has failed.

In general case, there will be no restrictions to the fault types inside the modules (some restrictions considered later may be introduced to simplify the fault diagnosis at the cost of worse diagnostic resolution).

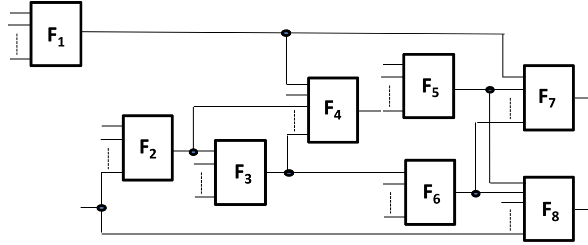


Figure 6.3 *A combinational circuit*

The first phase of the high-level fault diagnosis is carried out in the network of modules according to the cause-effect approach by using its pre-generated fault dictionary. The module-level fault dictionary is represented as the matrix $D = ||d_{ij}||$ where i denotes the number of test pattern, and j denotes the module $m_j \in M$ (a subcircuit or a library component in the network). The notation $d_{ij} = 1$ is used if the test pattern t_i may detect a faulty signal on the output y_j of the module m_j .

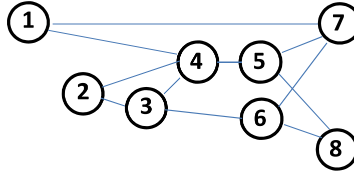


Figure 6.4 *A graph representing the combinational circuit in Figure 6.3*

Let $M_i = \{m_j\} \subset M$ be the subset of modules corresponding to the i-th row of the matrix D, so that $m_j \in M_i$ if $d_{ij} = 1$.

Consider a test set T to be used for fault diagnosis. Partition the set T in accordance with the results of test experiment into two subsets: $T = T_F \cup T_P$ where $T_F \cap T_P = \varnothing$, T_F is the subset of test patterns which failed during the test, and T_P is the subset of test patterns which passed. Denote by the vector:

$$M_F = \bigcup_{i:t_i \in T_F} M_i$$

the subset of all modules suspected as faulty on the basis of failing subset of test patterns T_F, and by the vector:

$$M_P = \bigcup_{i:t_i \in T_P} M_i$$

the subset of tested modules which have shown correct behavior on the basis of the subset of passed test patterns T_P.

Consider now the subset of modules $M^* = M_F - M_P$. It is easy to understand that if $M^* \neq \emptyset$ then the modules in M^* should be suspected as faulty to explain the failing of test patterns T_F .

Consider another subset of modules determined as $M^1_{cond} = M_F \cap M_P$. All these modules in M^1_{cond} should be treated as ambiguous, because they may explain the failing of test patterns T_F , however, not necessarily. Let call this subset as conditionally suspected modules after the first phase of high-level fault diagnosis. The total subset of modules suspected as faulty after the first phase of high-level diagnosis is determined as $M^1 = M^* \cup M^1_{cond}$.

Example 2. Table 6.1 represents an example of the module-level (high-level) fault dictionary D.

Table 6.1 *Module-level fault dictionary*

Test	Modules							
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
t_1	1		1			1	1	1
t_2	1						1	1
t_3			1			1	1	1
t_4		1	1			1	1	1
t_5				1	1		1	1

Assume the test patterns t_1 and t_2 failed during the test, e.g. $T_F = \{t_1, t_2\}$, and other three test patterns $T_P = \{t_3, t_4, t_5\}$ passed. Hence, $M_F = \{m_1, m_3, m_6, m_7, m_8\}$, and $M_P = \{m_2, m_3, m_4, m_5, m_6, m_7, m_8\}$. On this basis, the result of the first phase of high-level fault diagnosis is following: $M^* = \{m_1\}$ and $M^1_{cond} = \{m_3, m_6, m_7, m_8\}$. The modules m_3, m_6, m_7 and m_8 remain conditionally suspected as faulty.

In this phase of high-level diagnosis the information from the passed patterns $T_P = \{t_3, t_4, t_5\}$ cannot be used to reduce the subset of suspected faulty modules in M^1_{cond} , because there is no data in module-level fault dictionary which defects have been tested in modules M^1_{cond} during the test sets T_F and T_P .

6.3 Defect reasoning in modules by using the conditional stuck-at fault model

This section describes how the functional fault model consisting of pairs of SAF and defect conditions can be used for high-level defect reasoning in suspected faulty modules.

Consider a module $m \in M$ which is represented by a *Boolean function* $y = f(X)$, $X = (x_1, x_2, ..., x_n)$. Introduce a symbolic *Boolean variable* Δ for representing a given defect in the module, which converts the fault free function f into another faulty function f^{Δ}. Construct for this defect a generic parametric function:

$$y^* = f^*(x_1, x_2, ..., x_n, \Delta) = \overline{\Delta} f \vee \Delta f^{\Delta}$$

to model the defect of the module m as a function of the defect variable Δ, which describes jointly the behavior of the module for both, fault-free and faulty cases. For the faulty case, $\Delta = 1$, and for the fault-free case, $\Delta = 0$, i.e. $y^* = f^{\Delta}$ if $\Delta = 1$, and $y^* = f$ if $\Delta = 0$. The solutions $W_y(\Delta)$ of the Boolean differential equation

$$\frac{\partial f^*}{\partial \Delta} = 1 \tag{6.1}$$

describe the set of conditions (input signals of the module) which activate the defect Δ to produce an error on the output line y of the module. To find the conditions $W_y(\Delta)$ for a given defect Δ, the corresponding logic expression for the faulty function f^{Δ} has to be created, either by logical reasoning or by carrying out defect simulation directly, or by carrying out real experiments to learn the physical behavior of different defects. The described method represents a general approach to map an arbitrary transistor level physical defect inside the module m to the higher logic (or module) level.

Example 3. As an example, assume there is a short inside the transistor circuit in Figure 6.5 described by the function:

$$y = f(X) = x_1 x_2 x_3 \vee x_4 x_5$$

The short changes the function of the circuit as follows:

$$y = f^{\Delta}(X) = (x_1 \vee x_4)(x_2 x_3 \vee x_5)$$

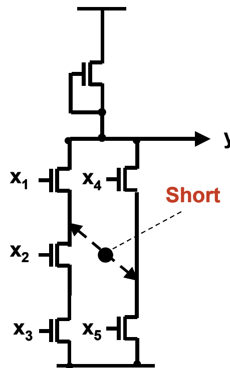


Figure 6.5 *Transistor circuit with a short*

Using the defect variable *Δ* for the short, a generic Boolean differential equation is created and solved as follows:

$$\frac{\partial y^*}{\partial \Delta} = \frac{\partial((x_1 x_2 x_3 \vee x_4 x_5)\overline{\Delta} \vee (x_1 \vee x_4)(x_2 x_3 \vee x_5)\Delta)}{\partial \Delta} =$$

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

From the equation three possible solutions follow: 10x01, 1x001, 01110 where x is do not care value. Each of them can be used as a test pattern for the given short, or used as the defect condition for SAF at the output *y*. According to the definition of the functional fault model for the defect *Δ* , the input conditions are $W_y(\Delta) = \{10x01, 1x001, 01110\}$.

The full set of conditions $W_y = \{W_y(\Delta)\}$ is called as the *functional fault model* to represent all the physical defects through functional deviations in the behavior of the module *m*: a physical level defect *Δ* produces a higher logic level erroneous signal on the module output *y* if $W_y(\Delta) = 1$ [69] [70].

By using the set of conditions W_y it is possible to map the defects from lower physical level to higher logic level for fault simulation purposes or vice versa, to map the faulty logic signals from the module-level to physical defects for fault diagnosis purposes. If the modules of the circuit represent standard library components (e.g. complex gates) the described analysis for finding conditions should be made once for all library components, and the sets of calculated conditions W_y will be included in the form of defect dictionaries into the library of components.

6.4 Effect-cause high-level fault reasoning

The effect-cause high-level fault reasoning as the second phase of high-level fault diagnosis is defined as follows. The set of all conditional candidate faulty modules $M^1_{cond} = M_F \cap M_P$ found in the first high-level diagnosis phase is taken, and the sets of fault conditions $X_{F,j}$ for all modules $m_j \in M^1_{cond}$ suspected conditionally are created. The set M^1_{cond} can be interpreted as the effects evoked locally by the possible defects in the modules $m_j \in M_F \cap M_P$. In these suspected modules indirect defect reasoning has to be carried out either to determine the possible defect causes or to remove the not faulty modules from suspicion. The task of the second phase of high-level fault diagnosis is to locate the faulty module or to reduce the set of suspected faulty modules as much as possible.

Denote by X_{ij} the local input condition of the module m_j at the test pattern t_i . Then group all these input conditions for all the modules in such a way that $X_{F,j}$ contains all the local input conditions of the module m_j during the test patterns,

which tested the module and failed, and $X_{P,j}$ contains all the local input conditions of the module m_j during the test patterns, which tested the module and passed.

Each condition $w_F \in X_{F,j}$, so that $w_F \in W_j(\Delta)$, refers to a defect Δ which may have been detected by the test patterns in T_F, and therefore should be suspected because all the test patterns in T_F failed. On the other hand, each input condition $w_P \in X_{P,j}$, so that $w_P \in W_j(\Delta)$, refers to a defect Δ which should have been detected by the test patterns in T_P. However, since all the test patterns in T_P passed, the defect Δ cannot be any more suspected.

From the above reasoning it results that only these defects in the module m_j which are activated by the conditions $w \in (X_{F,j} - X_{P,j}) \cap W_y$ may be suspected. Hence, a module $m_j \in M^1_{cond}$ can be suspected as faulty only if

$$(X_{F,j} - X_{P,j}) \cap W_y \neq \varnothing \quad (6.2)$$

The reasoning proceeds in the following way. From all conditions $X_{F,j}$ where $m_j \in M^1_{cond}$ are deleted the conditions $X_{P,j}$ trying to suppress the set M^1_{cond}. The result of the second phase of high-level fault diagnosis will be the reduced set of candidate faulty modules $M^2 = M^* \cup M^2_{cond}$, where $M^2_{cond} \subseteq M^1_{cond}$, and for all $m_j \in M^2_{cond}$ the condition (6.2) is valid.

Example 4. Consider again the example in Table 6.1 where the first two test patterns failed during the test experiment, and the other three passed. Let the module m_3 had the local input vector $X_{11} = (1011)$ during the test pattern t_1 which tests the module, according to Table 6.1. Assume here that the module has 4 input lines. Since t_1 failed, it means that in the module m_3, a defect Δ, activated by the conditions $w_F = (1011) \in W_3(\Delta)$ on the inputs of the module, should be suspected. On the other hand, let the same module had on its inputs the same condition $w_P = w_F = (1011)$ during the test pattern t_4 which tested the module m_3 as well, but passed. Since the test pattern t_4 activated the same defect Δ which was under suspicion after the test pattern t_1, there is a contradiction, and the defect Δ cannot be present. So, the module m_3 can be removed from the set of suspected modules $M^1_{cond} = \{m_3, m_6, m_7, m_8\}$. After such a procedure of the indirect effect-cause reasoning of defects on the module-level, in the ideal case $M^2_{cond} = \varnothing$ may be obtained. In this case, the result of the high-level fault diagnosis shows that a single module, $M^2 = M^* = \{m_1\}$, should be suspected as faulty.

In the case of multiple faulty modules, fault masking may lead to equivalent conditions on modules in the passing and failing cases. As the consequence, too many suspected modules may be pruned. To avoid such mistakes, several methods for ranking suspected modules in M^1_{cond} can be proposed. For example, the more contradictions will be used for deleting the conditions from $X_{F,j}$ of a module m_j, the less is the probability that all of the deletions were mistakes because of fault masking, and consequently, the less is the probability that the module removed from M^1_{cond} is still faulty.

6.5 Low-level fault diagnosis

Low-level fault diagnosis is carried out in the modules determined during the high-level reasoning as faulty. It may be carried out either by cause-effect diagnosis based on the defect dictionaries pre-computed for the library components, or by effect-cause diagnosis inside the modules if the defect libraries are not available.

Consider as an example the fault dictionary in Table 6.2 for a library complex gate AND2,2/NOR2 [96] regarded as a module in the circuit under diagnosis. In the table, 25 different defects (shorts between inputs or internal nodes of the module) are considered with corresponding names in column 2 and erroneous functions in column 3. Other columns form the defect dictionary and correspond to the input patterns of the 4-input module (given with decimal numbers). The entries “1” in columns show which defects are tested by which patterns.

Table 6.2 *Defect library for a gate AND2,2/NOR2* [96]

k	Fault Δ_k	Erroneous function $f^{\Delta k}$	Local input test patterns w_j															
			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	A/C	not((A*C)*(B+D))				**1**				1					**1**	1		
2	**A/D**	**not((A*D)*(B+C))**				**1**				1					**1**		1	
3	A/N1	not(B*(not(A)+C+D)+C*D)					1	1	1						**1**			
4	A/Q	A*(not(C*D))	1	1	1		1	1	1						**1**	1	1	
5	A/gnd	not(C*D)													**1**	1	1	
6	A/vdd	not(B+(C*D))					1	1	1									
7	B/C	not((B*C)*(A+D))				**1**								1	**1**	1		
8	**B/D**	**not((B*D)*(A+C))**				**1**								1	**1**		1	
9	B/N1	not(A+C*D)									1	1	1					
10	B/Q	B*(not(C*D))	1	1	1						1	1	1		**1**	1	1	
11	B/gnd	not(C*D)													**1**	1	1	
12	B/vdd	not(A+(C*D))									1	1	1					
13	C/N1	Not((A+B+D)*(A*B+not(C)+D))		1			1	1			1	1						
14	C/Q	C*(not(A*B))	1	1		**1**	1	1		1	1	1		1				
15	C/gnd	not(A*B)				**1**				1				1				
16	C/vdd	not(D+A*B)		1				1				1						
17	D/N1	not(A*(B+C+notD)+B*notD+not((A+B)*D)			1		1		1		1		1					
18	D/Q	D*(not(A*B))	1		1	**1**	1		1	1	1		1	1				
19	D/gnd	not(A*B)				**1**				1				1				
20	D/vdd	not(C+A*B)			1				1				1					
21	N1/Q	not(A*B*C*D)				**1**				1				1				
22	N1/gnd	sa-0 for Q	1	1	1		1	1	1		1	1	1					
23	N1/vdd	not(C*D)													**1**	1	1	
24	Q/gnd	sa-0 at Q	1	1	1		1	1	1		1	1	1					
25	Q/vdd	sa-1 at Q				**1**				1				1	**1**	1	1	1

Assume that as the result of the test experiment, for this particular module m_j has been found that $X_{F,j} = \{w_3, w_{12}\}$, and $X_{P,j} = \{w_0, w_5, w_8, w_{10}, w_{13}, w_{15}\}$. The input patterns not exercised during the test are highlighted by grey colour. It is easy to see that the defects Δ_2 and Δ_8 should be suspected as present. For better diagnostic resolution, to distinguish these defects, additional input patterns should be used for testing, e.g. w_7 or w_{11} .

If no diagnosis is possible with using the defect dictionary, for example when the real existing defect is not covered by the dictionary, or in the case when the defect dictionary for a module is missing, the low-level effect-cause defect reasoning inside the module is needed. This can proceed for example by the method of critical path tracing.

6.6 Experimental data

The goal of the experimental research was to evaluate the achievable high-level module-based diagnostic resolution (Table 6.4) and to compare it with traditional *stuck-at fault* (SAF) based diagnostic resolution (Table 6.5). Experiments were carried out with ISCAS'85 [92] [93], ISCAS'89 [95] and ITC'99 [97] [98] benchmark circuits (Table 6.3). All the circuits were presented as networks of modules where as modules the *fanout-free regions* (FFR) were selected. Three different types of tests were used for evaluating the diagnostic resolution (Table 6.4): long pseudorandom test sequences, short deterministic test sequences, and combination of both sequences. The pseudorandom test sequences were generated by LFSR based test generator and optimized by selecting proper seeds and polynomials. The LFSR was stopped when no useful test patterns were found in a reasonable time. The pseudorandom test was selected to have more test patterns for detecting the same fault with the goal to improve the distinguishability of the faults. The deterministic test patterns were synthesized by a genetic test generator. The quality of test patterns, i.e. the maximum possible test coverage was not the target. The goal was not to generate the best test sequences, rather to evaluate and compare the diagnosability, i.e. diagnostic resolutions for two methods and for a given test sequence. To generate diagnostic tests with as good as possible diagnostic resolution is the task not considered in this section.

Table 6.3 *Characteristic data of benchmark circuits*

Circuits	Inputs	Outputs	Nodes (lines)	Modules	Complexity reduction
c1908	33	25	1394	248	11.2
c2670	233	140	2075	430	9.7
c3540	50	22	2784	378	14.7
c5315	178	123	4319	633	13.6
c7552	207	108	5795	920	12.6
s9234	247	250	5597	1263	8.9
s13207	700	790	12441	2014	12.4
s15850	611	684	14841	2202	13.5
s35932	1763	2048	32624	7343	8.9
b12_C	126	127	1000	512	3.9
b14_C	277	299	19491	2708	14.4
b15_C	485	519	18248	2872	12.7
Average					11.4

In Table 6.3, the characteristic data of the benchmark circuits are presented. In columns 2 - 4, the numbers of inputs, outputs and nodes (lines) are presented, respectively. The number of lines indicates the number of possible stuck-at faults in the circuit. Column 5 lists the number of modules, and in column 6 the complexity reduction in the module-level network is shown compared to the gate-level network. The complexity reduction is calculated as follows: number of lines is multiplied by 2, since SA0 and SA1 fault per line can present, and divided by number of modules where for each module is used 1-bit notation: faulty or fault free. The reduction in average 11.4 times reflects the decrease of the complexity in fault dictionaries in case of using FFR-modules instead of gates and SAF lists. If the obtained complexity reduction is not satisfied, which is especially actual for *very-large-scale integration* (VLSI) designs, the initial FFR-modules can be grouped into larger modules to reduce the size of high-level fault dictionary at the cost of diagnostic resolution.

In Table 6.4, the results of diagnostic resolution for three different test types are presented. In each section, the characteristics of tests are given: test length and stuck-at fault coverage (SAF %), and the average diagnostic resolutions (the number of suspected modules) after the first and the second phases of high-level diagnosis are depicted.

Table 6.4 *Diagnostic resolution for high-level module-based diagnosis*

Circuits	Pseudorandom test				Deterministic test				Both tests			
	Test length	SAF %	Average resolution		Test length	SAF %	Average resolution		Test length	SAF %	Average resolution	
			Phase1	Phase2			Phase1	Phase2			Phase1	Phase2
c1908	4420	99.48	86.6	1.42	121	99.48	110.6	2.08	4541	99.48	86.1	1.41
c2670	22682	88.65	223.9	3.45	72	88.46	235.8	4.39	22754	90.36	221.0	3.24
c3540	9631	95.54	69.6	2.53	155	95.54	85.7	3.89	9786	95.54	69.6	2.51
c5315	1793	98.89	213.2	1.85	119	98.89	238.4	3.45	1912	98.89	212.0	1.84
c7552	24337	94.08	287.5	2.60	188	95.18	304.0	2.90	24525	95.52	285.7	2.58
s9234	29873	86.21	271.3	6.32	365	92.19	398.0	6.75	30238	92.23	205.0	4.50
s13207	29694	96.81	781.3	10.53	434	98.19	1035.5	9.98	30128	98.19	701.6	7.60
s15850	29360	90.99	669.4	10.50	370	94.19	886.0	10.45	29730	94.22	557.3	4.12
s35932	168	90.81	3748.4	63.85	55	88.49	4391.1	17.87	223	90.81	3554.5	6.45
b12_C	27205	98.33	151.4	4.15	155	99.77	239.6	5.05	27360	99.83	126.8	2.83
b14_C	29388	88.86	249.8	18.29	848	92.76	655.7	6.15	30236	95.69	531.7	2.69
b15_C	39198	86.56	756.5	14.06	498	88.77	1069.1	17.67	39696	94.74	929.1	11.08
Average			**625.7**	**11.6**			**804.1**	**7.55**			**623.3**	**4.2**

Due to large circuits (number of possible stuck-at faults) and long pseudorandom test sequences only one randomly selected stuck-at fault per each module was injected and simulated. Thus, the number of simulated SAFs equals to the number of modules (Table 6.3). The pseudorandom test was selected to have more test patterns for detecting the same fault with the goal to improve the distinguishability of the faults and also to meet demands of embedded fault diagnosis. However, the desired effect of applying pseudorandom test sequences for distinguishing the faults has not been achieved when comparing the final

diagnostic resolutions of pseudorandom and deterministic test sequences. The difference of results after the Phase 2 is not significant although the test lengths of pseudorandom sequences for some circuits are hundred times longer. The average diagnostic resolution has been improved when the combined test sequences consisting of pseudorandom and deterministic patterns were applied. For example, for large circuit b14_C the diagnostic resolution after Phase 2 was 18.29 and 6.15 for pseudorandom and deterministic tests, respectively. The combined test enhanced the accuracy of diagnosis to 2.69.

According to Table 6.4 the Phase 2 has reduced the average number of suspected modules in a case of SAF considerably compared to Phase 1. Thus, the diagnostic data of passed patterns are playing an important role when suppressing candidate faulty modules by the procedure of deleting failed conditions by passed conditions. For the combined test and over all circuits the average number of suspected modules 4.2 is quite low. Taking into account the structural locations of suspected modules, and generating tests dedicated for diagnostic purposes, the diagnostic resolution can be further reduced.

Table 6.5 shows the comparison results of the proposed module-level based diagnosis with traditional SAF-based diagnosis obtained from experiments performed on ISCAS'85 benchmark circuits using the same pseudorandom tests. The values in column 4 depict the average number of suspected SAFs as a result of fault diagnosis over all tested SAF by given test sequences, whereas column 5 shows the average number of modules where these suspected SAFs were located. The values in column 6 depict the results of module-based approach that in average are 2.4 / 1.4 = 1.7 times worse compared to SAF-based approach. The difference in diagnostic resolution is due to that SAF-based method directly targets SAFs while the proposed approach identifies candidate faults with accuracy of module locations. On the other hand, the module-based approach reduces fault dictionary complexity significantly, in average by 12.3 times; although the obtained diagnostic resolution 2.4 remains quite high that indicates the feasibility of the proposed module-based approach.

Table 6.5 *Comparison of SAF-based and module-based diagnosis*

Circuits	Test characteristics		Diagnostic resolution			Fault dictionary complexity reduction
	Length	SAF %	SAF-based		Module-based	
			saf	modules	modules	
c1908	4420	99.48	2.9	1.2	1.4	11.2
c2670	22682	88.65	4.1	2.0	3.5	9.7
c3540	9631	95.54	2.2	1.2	2.5	14.7
c5315	1793	98.89	2.2	1.1	1.9	13.6
c7552	24337	94.08	2.6	1.5	2.6	12.6
Average			**2.8**	**1.4**	**2.4**	**12.3**

6.7 Conclusions

- The main drawbacks of the traditional cause-effect diagnosis methods are the dependency on the fault model and poor scalability of fault dictionaries.
- A novel method for high-level fault diagnosis based on module-level fault dictionaries is developed. No fault models are used, and the objective of diagnosis is faulty module in a given network of modules. As modules, either library components (e.g. complex gates) of digital circuits or arbitrary subcircuits are considered. The method combines cause-effect and effect-cause strategies. The approach is scalable, and helps to cope with the growing complexities of digital circuits. No restrictions to fault or defect multiplicity inside the modules are set.
- In addition, a novel hierarchical approach is developed, which combines the high-level module-oriented and low-level defect-oriented reasoning to improve the diagnostic resolution in the case when information about possible defects in modules is available. Two methods are proposed for locating defects in candidate faulty modules: cause-effect reasoning based on defect libraries of modules, and effect-cause reasoning inside the modules.
- The complexity of the proposed method compared to the flat logic level and fault model based diagnosis is reduced. The size of the high-level fault dictionary for the whole circuit depends linearly only on the number of modules to be determined as faulty or not faulty, and not on the number of possible faults or defects as traditionally. The size of the modules, however, will be the trade-off between the complexity of the high-level dictionary and the diagnostic resolution.
- The complexity of the diagnosis problem is reduced in average 11.4 times for the case of FFR-based modules compared to gate-level diagnosis. The diagnostic resolution in the module-based diagnosis is slightly worse than in the case of SAF-based fault diagnosis due to significant reduction of diagnostic data stored in module-based dictionary.
- The high diagnostic resolution at the module-level was achieved by implementing a novel high-level effect-cause reasoning based on the concept of functional fault model (conditional SAF model). This concept is based on indirect mapping of physical defects from transistor level to module-level, so as to carry out indirect defect based fault reasoning at higher module-level.

Chapter 7
IMPLEMENTATION OF THE RESEARCH RESULTS IN THE TEACHING ENVIRONMENT

This chapter presents a multifunctional e-learning environment with remote access for learning, getting hands-on experience, and carrying out laboratory research in developing optimized procedures for locating faults in complex electronic systems. It is a combination of a collection of software tools which simulate a system under diagnosis, emulate a pool of different strategies, methods, and algorithms of diagnostic reasoning and fault location, and allow to experiment with different embedded self-diagnosing architectures. Hands-on experiments target research teaching issues. The interactive character of the environment makes experiments attractive and helps to raise the students' curiosity.

7.1 Overview of fault diagnosis problems

There are two basic problems related to the engineering fields of verification and testing of technical systems: test program generation and failure diagnosis [8] [99]. Diagnosis is the most time-consuming process in the digital hardware design cycle. The causes of failures may be design errors or physical defects called also faults. Design error diagnosis is necessary when the design description fails testing due to bugs introduced either by designers or by design tools. Fault diagnosis is used after a fabricated chip fails testing due to defects in the

manufacturing process. The objective of fault diagnosis is to determine the cause of failure in a system, circuit or chip.

There are two general approaches to fault diagnosis: fault model based diagnosis and model-free fault diagnosis. The first one is a traditional approach. Modern diagnosis approaches are model-free processes that identify the faulty or erroneous areas in a circuit or system without assuming any specified fault lists or error models.

Diagnosis techniques are usually classified into cause-effect and effect-cause techniques. Cause-effect diagnosis, also known as dictionary-based diagnosis, usually pre-compiles the failing responses of all modeled faults for a design and stores them in a dictionary. Effect-cause diagnosis identifies initial candidates by simulating failing input vectors followed by a reasoning process.

As process technologies shrink and designs become more complex, *built-in self-test* (BIST) is gaining increasing acceptance as an industry-wide test solution, because it provides a low-cost solution to both test generation and test application.

Both, model based and model-free approaches as well as cause-effect and effect-cause techniques are supported in the presented environment. Also different embedded BIST and self-diagnosis architectures are emulated to evaluate the efficiency of diagnosis.

7.2 Description of the environment

In the following a new laboratory tool *Diagnozer* for investigation, modeling and simulation of diagnostic processes for locating faults in complex electronic systems is presented. *Diagnozer* is a part of the tool set called *BIST Analyzer & Diagnozer* [100].

BIST Analyzer (BISTA) [101] is a training and research tool for learning basic and advanced issues related to PRPG-based test pattern generation. *Linear feedback shift registers* (LFSR) and other *pseudo-random pattern generators* (PRPG) have become one of the central elements used in test and self-test of contemporary complex electronic systems like processors, controllers, and high-performance integrated circuits. Unlike other similar systems, this tool facilitates study of various test optimization problems, allows fault coverage analysis for different circuits and with different LFSR parameters. The main didactic aim of the tool is presenting complicated concepts in a comprehensive graphical and analytical way. The multi-platform JAVA runtime environment allows for easy access and usage of the tool both in a classroom and at home. The BISTA represents an integrated simulation, training, and research environment that supports both analytic and synthetic way of learning.

The *Diagnozer* environment consists of the following three components:

(1) object under investigation (the model of a system to be diagnosed)

(2) diagnostic environment (a software for modeling the behavior of diagnostic hardware/software tools to be used for fault location in the system)

(3) set of strategies, methods and algorithms to control the fault location procedures with available diagnostic tools

The user interface of the system is illustrated in Figure 7.1. Different test generators can be chosen, different stored models of systems under diagnosis can be imported, the interface between system and diagnostic analysis block is the objective of research, design and optimization.

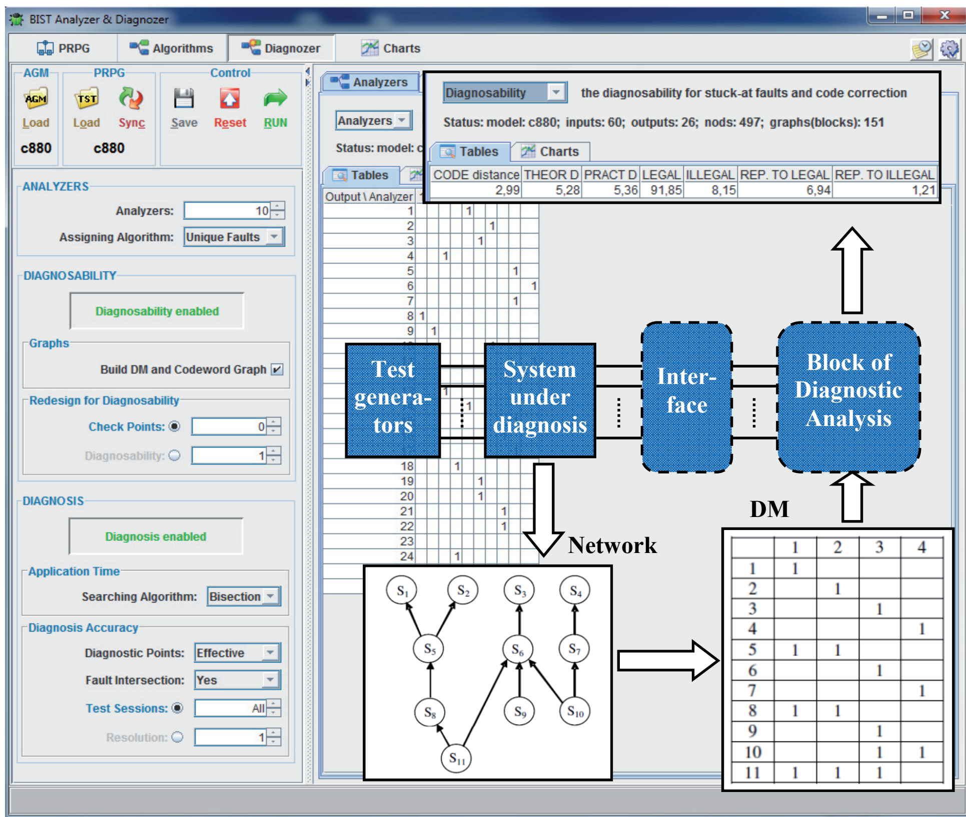


Figure 7.1 *User interface of the diagnosis environment*

The model of the system (object under investigation) can be imported from the *electronic design interchange format* (EDIF) or *VHSIC hardware description language* (VHDL) formats and can be represented either in gate-level or module-level. The tools for diagnostic modeling of the given system support two different approaches for fault diagnosis:

- the classical *stuck-at-fault* (SAF) model based diagnosis
- a novel fault model free diagnosis approach where the goal of diagnosis is to locate the faulty module

To investigate the embedded self-diagnosis architectures and to analyze their properties and efficiencies, the following methodical approaches and means are addressed: methods for generating the set (or sequence) of test patterns and methods for observing and processing the response data.

The test generation tools include [101]:

- deterministic, genetic or random test generators for producing test sequences to be stored in the tester memory
- pseudorandom test generators for producing on-line test sequences
- hybrid BIST architectures implementing the store-and-generate concept

Test response analysis possibilities. The test response can be fixed:

- by direct capturing and storing of all the responses
- by processing the responses with single *signature analyzer* (SA)
- by processing the responses with multiple SAs

The following architectures for signature analysis can be used (Figure 7.2):

- Multiple Input SAs (MISR) where all the outputs of *circuit under test* (CUT) are connected to a single *multiple input signature register* (MISR)
- a set of MISR-s where each MISR is connected to a different subset of outputs of CUT
- single output SAs where for each output of the CUT a dedicated SA is available

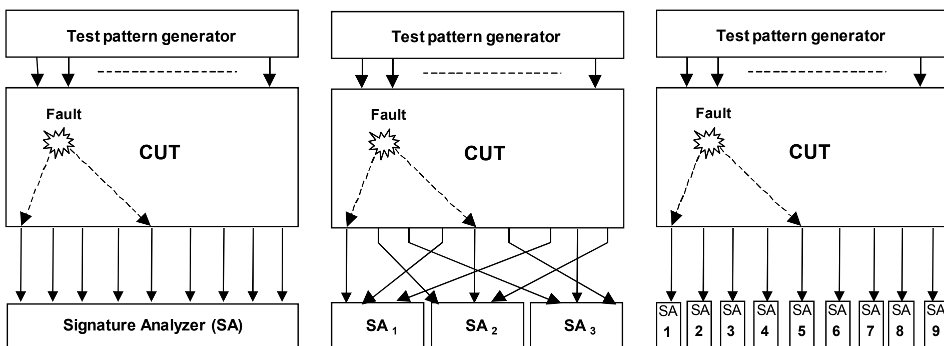


Figure 7.2 *Architectures for signature analysis*

There are tools to design and optimize the interface between the outputs of the system and the block of multiple signature analyzers with the goal to achieve the best possible diagnostic resolution.

The structure of the system under diagnosis will be analyzed, and the diagnostic model as a network of components will be created and represented as a diagnostic matrix to be used in fault diagnosis, evaluation of the diagnosability, and redesign of the system for better diagnosability.

Four strategies of fault diagnosis are implemented: binary bisection of test patterns, binary bisection of detected fault sets, doubling and jumping algorithms. The diagnostic environment includes different possibilities for selecting *diagnostic points* (DP) in test sequences, different possibilities to define the breakpoint for diagnostic procedure and different options to identify a set of suspected faults among failing patterns.

Using this environment the following problems of diagnosis can be addressed in the laboratory:

- methods and strategies of organizing diagnostic experiments
- methods for optimization of diagnostic procedures
- methods for improving the diagnostic resolution
- methods for finding trade-offs between the time cost of fault diagnosis (length of the diagnostic procedure) and the diagnostic resolution
- methods for evaluating the diagnosability of systems
- methods for improving the diagnosability of digital systems by redesigning

7.3 Laboratory research scenarios

In the following a series of three laboratory research scenarios are described which are based on the presented diagnostic environment:

1. *Diagnostic circuitry*: design and optimization of the built-in response analyzing circuitry
2. *Diagnostic algorithms*: investigation of the properties of different diagnostic strategies
3. *Object of diagnosis*: evaluation and design for diagnosability of digital systems

Inputs files for *Analyzers*, *Diagnosability* and *Diagnosis* tools are the following:

- *Structurally Synthesized Binary Decision Diagram* (SSBDD) model format (.agm) – model of a circuit

- test vectors format (.tst) – sequence of test patterns with corresponding fault table for given circuit

Output data (results) of the tool are presented either graphically or in the form of tables that can be saved in .csv format (text delimiter is semicolon ";") for comfort and easy reading using *Microsoft Excel* or *OpenOffice*.

The presented scenarios are adapted for both analytic and synthetic study, where the students first learn the subject by observation (using prepared examples) and then generate and/or solve their own specific exercises. The scenarios cover various strategies and methods of organizing and optimizing test generation and fault diagnosis.

7.3.1 Research Scenario 1: Diagnostic Circuitry

***Goals of the work*:**

Investigations of the properties of the response analyzing circuitry based on partitioning a single signature analyzer into a set of analyzers. Examples of the architecture of the response analyzing circuitry are shown in Figure 7.2, representing the *circuit under test* with a fault, and a block of response analyzers implemented as multiple input *signature analyzers* (see Chapter 4).

***Description of the research environment and the functionality of tools*:**

Functions of the tool:

- The main function of the tools used in this research is to assign in an optimal way the selected number of signature analyzers to the outputs of the circuit under diagnosis, using different optimization algorithms

Workflow for the research scenario is depicted in Figure 7.3:

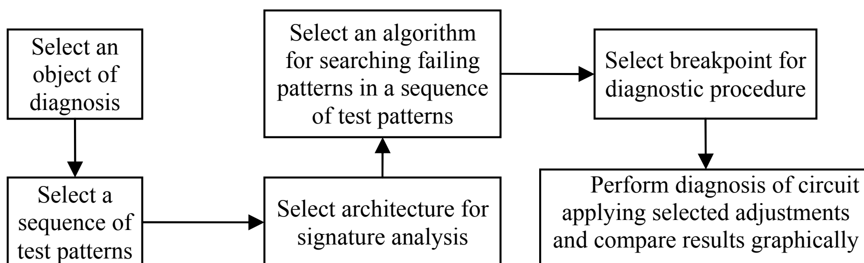


Figure 7.3 *Workflow for research scenarios 1 and 2*

Options (*ANALYZERS* panel in Figure 7.1):

- **Analyzers:**
  - 1…10 – number of applied signature analyzers

- **Assigning algorithm:**
    - *Unique Faults.* First, finds a unique set of faults for each output (faults detected only by this single output) and sorts them in decreasing order. Second, takes the output from the sorted set and assigns it to an analyzer in such way that unique faults would be equally distributed between analyzers.
    - *Equal Subsets.* First, to each SA an initial output of CUT is assigned. Second, on each step an arbitrary output of CUT is selected, and a SA is found as the best solution to be connected to the selected output of CUT in a such way that the diagnostic resolution would be equal for all subsets D_j . Ideally, $D_j = | F | / (2^m - 1)$ where $| F |$ – number of possible faults and m – number of SAs.

Output data of the tool (*Analyzers* tab in Figure 7.1):

- a table "*Analyzers*" where is shown which outputs are assigned to which analyzers if the number of analyzers is selected greater than 1

***Tasks of the work*:**

1. Compare the dependency of the diagnostic resolution on the number of analyzers for the given method of fault diagnosis and for the single first failed test pattern. Create the curves or diagrams of this dependency. An example of such diagram is depicted in Figure 7.4 where circuits c2670, c5315 and c7552 are considered and the diagnostic resolution, obtained after the first failing patterns has been found, is compared for different number of SAs.

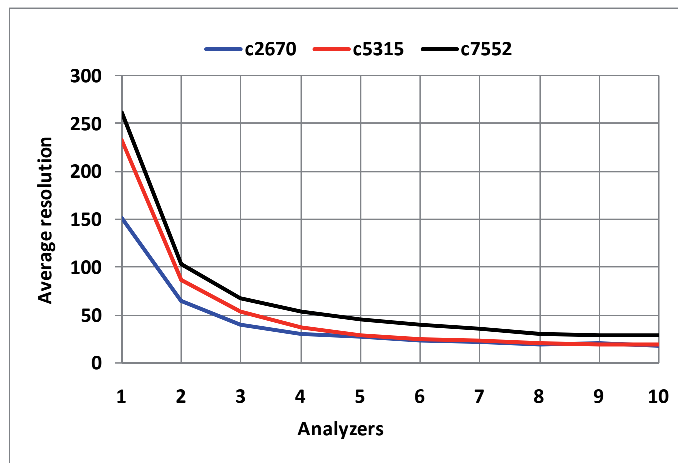


Figure 7.4 *Dependency of the resolution on the number of SAs for Task 1*

2. Compare the dependency of the diagnostic resolution on the number of analyzers for the given method of fault diagnosis and for the full diagnostic procedure, i.e. for all failed test patterns. Create the curves or diagrams of this dependency. An example of such diagram is depicted in Figure 7.5.

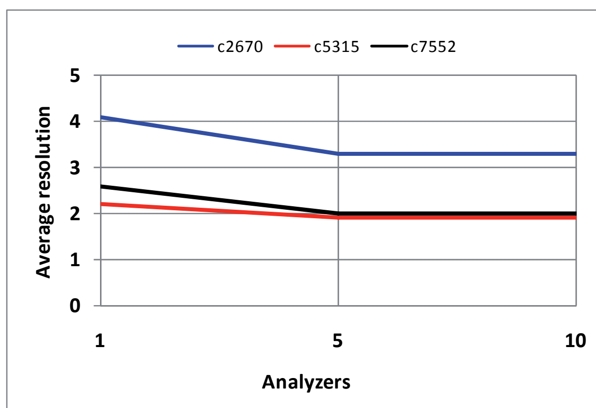


Figure 7.5 *Dependency of the resolution on the number of SAs for Task 2*

3. Compare the effectiveness of assigning algorithms based on the results obtained in previous tasks.

7.3.2 Research Scenario 2: Diagnostic Algorithms

***Goals of the work*:**

Investigations of the properties of different methods and strategies of fault diagnosis: binary bisection of patterns, binary bisection of faults, jumping and doubling (see Chapter 3).

***Description of the research environment and the functionality of tools*:**

Functions of the tool:

- The tool simulates injection of each stuck-at fault tested by given sequence of test patterns and performs fault diagnosis using following options

Workflow for the research scenario is depicted in Figure 7.3.

Options (*DIAGNOSIS* panel in Figure 7.1):

- Diagnosis application time (number of test sessions):
    - **Searching Algorithm:**
        - *Binary bisection of faults*

        - *Binary bisection of test patterns*
        - *Doubling*
        - *Jumping*
- Diagnosis Accuracy (diagnostic resolution):
    - **Diagnostic points** – test patterns that are used for fault diagnosis from a test sequence
        - *All patterns* – time-consuming when a very long sequence of pseudorandom test patterns
        - *Effective* – patterns that detect new faults compared to previous patterns in a given sequence of test patterns
        - *Effective/2* – each second of effective patterns is used
        - *Effective/4* – each fourth of effective patterns is used
        - *Effective/8* – each eighth effective patterns is used
    - **Fault intersection**. For instance, if there are two failed patterns t_1 and t_2 that detect faults $F(t_1)$ and $F(t_2)$, and two signature analyzers SA_1 and SA_2 , where $SA_1(t_1)$ fails and $SA_2(t_1)$ passes, whereas $SA_1(t_2)$ and $SA_2(t_2)$ both fail. Following equations $F(t_1) = SA_1(t_1) \cup SA_2(t_1)$ and $F(t_2) = SA_1(t_2) \cup SA_2(t_2)$ are valid.
        - *Yes* (single fault assumption) – the set of suspected faults equals $D = (SA_1(t_1) \cap SA_1(t_2) \cap SA_2(t_2)) - SA_2(t_1)$
        - *No* (multiple fault assumption) – $D = SA_1(t_1) \cup F(t_2)$
        - *Half* (multiple fault assumption) –
          $D = SA_1(t_1) \cup (SA_1(t_2) \cap SA_2(t_2))$
    - **Test sessions** – number of failed patterns found and used for fault diagnosis:
        - 1…10 – diagnostic procedure is finished when given number of failed patterns is found or the end of test sequence is reached
        - *All* – diagnostic procedure proceeds until all failing patterns are found
    - **Resolution**:
        - 1…10 – diagnostic procedure proceeds until selected diagnostic resolution is achieved or all failing patterns are found

From two options *"Test sessions"* and *"Resolution"* only one can be chosen at a time.

Output data of the tool (*Diagnosis* tab in Figure 7.1):

- tables *"Tests"* and *"Resolution"* where for each stuck-at fault is presented the number of tests used for diagnosis and the achieved diagnostic resolution, respectively
- a curve *"Tests"* where is illustrated how many SAFs are tested by given or less number of tests, and a curve *"Resolution"* where is shown how many SAFs have been diagnosed by given or better diagnostic resolution
- a table *"Diagnosis"* where minimum, average and maximum number of tests and diagnostic resolution over all tested SAFs is presented
- a histogram *"Frequency"* where is illustrated frequencies of detections different SAFs by given sequence of test patterns

***Tasks of the work*:**

1. Calculate the cumulative numbers of all faults *F* diagnosed by *N* or less test sessions for different methods of diagnosis. Create the curves of these dependencies for all the investigated methods. Compare the efficiency of the methods. An example of such curve is depicted in Figure 7.6.

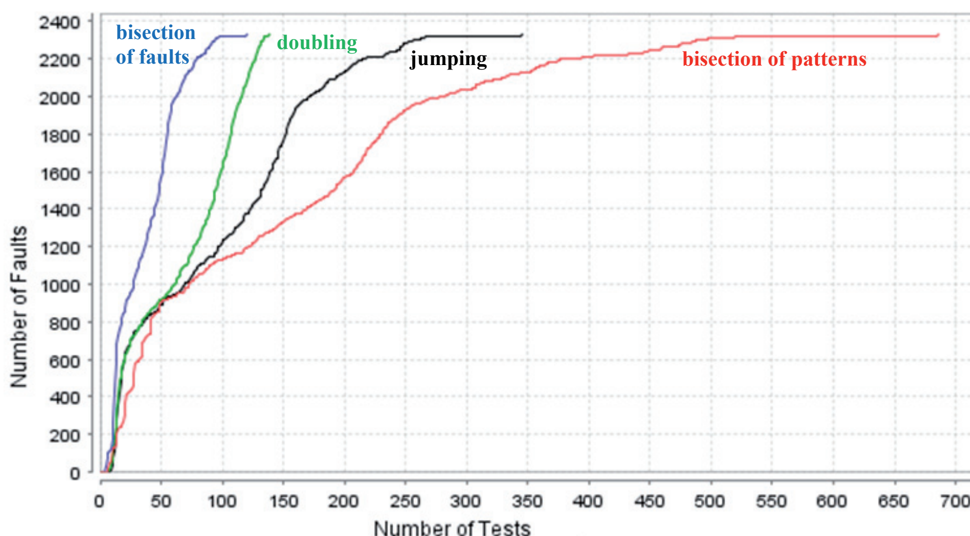


Figure 7.6 *Comparison of different diagnosis methods for circuit c2670*

2. Calculate the diagnostic resolution as the function of test length that is proportional to the number of failing patterns detected during the fault diagnosis. An example is depicted in Figure 4.4.
3. Compare the dependency of the diagnostic resolution and test length on the number of selected DPs for given method. Create the curves or diagrams of this dependency.
4. Perform a number of experiments with different adjustments so as to find out an optimal trade-off between the number of SAs, the time cost (test length) and the accuracy (resolution) of the fault diagnosis. An example is depicted in Figure 4.5.

7.3.3 Research Scenario 3: Object of Diagnosis

***Goals of the work*:**

Investigations of the potentials of the fault model free diagnosis by comparing it with classical stuck-at fault model diagnosis. Getting acquainted with conceptions of the diagnostic matrix of the circuit network, the graph of codeword distances, the method of rectification of diagnostic responses, and optimized redesign of the circuit for better diagnosability (see Chapter 5).

***Description of the research environment and the functionality of tools*:**

Functions of the tool (*DIAGNOSABILITY* panel in Figure 7.1):

- The tool builds up the *Diagnostic Matrix (DM)* and the *Codeword Graph.* Then evaluates the diagnosability of given circuit and performs redesigning of the circuit for better diagnosability according to selected criteria.

Workflow for the research scenario is depicted in Figure 7.7.

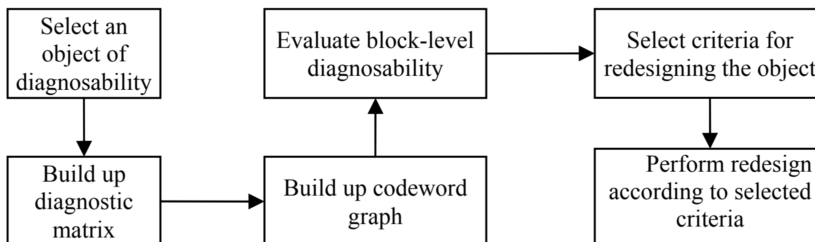


Figure 7.7 *Workflow for research scenario 3*

Options:

- Graphs
  - **Build DM and Codeword Graph**
- Redesign for Diagnosability
  - **Check Points**
    - 1...10 – number of check points to be inserted into a circuit for better diagnosability
  - **Diagnosability**
    - 1…10 – the diagnostic resolution to be achieved as a result of redesigning the circuit

From two options *“Check Points”* and *“Diagnosability”* only one can be chosen at a time.

Output data of the tool (*Diagnosability* tab in Figure 7.1):

- a table *“DM outs”* (or *“DM SAs”*) where is presented *diagnostic matrix DM* that shows the blocks (modules) connections to the outputs (or SAs)
- a table *“Codewords outs”* (or *“Codewords SAs”*) where is presented a graph of codewords in case of observing outputs directly (or via SAs)
- a table *“DM Groups outs”* (or *“DM Groups SAs”*) where is presented number of different codewords (groups) and average diagnosability with accuracy of the block (module) locations in case of observing outputs directly (or via SAs)
- a table *“Diagnosability”* where is presented average bit distance between codewords and diagnosability measures for stuck-at faults as a result of diagnosis with accuracy of block locations and adjusting the test responses
- a table *“Distortions”* which presents bit distance probabilities between the real test response (codeword) and the expected codeword over all stuck-at faults tested in the circuit
- a table *“Redesigned DM”* which presents the updated diagnostic matrix where selected number of checkpoints is inserted into the circuit for better diagnosability

***Tasks of the work*:**

1. Calculate the values of diagnosability measures for the given set of circuits.

2. Redesign the circuit by inserting optimal diagnostic *check points* into given circuit. Create two curves of average diagnosability as the function of the number of inserted check points: (1) for the case of randomly inserted check points, and (2) for the case of the optimal insertion of check points. In Figure 7.8 "worst case" and "best case" scenarios of average block-level diagnosability improvement are represented for circuit c1355.

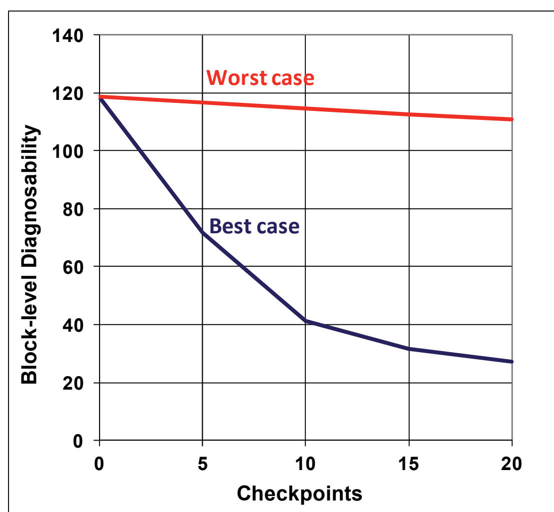


Figure 7.8 *Diagnosability improvement range for circuit c1355*

7.4 Conclusions

The presented conception of the environment for carrying out different research scenarios to get hands-on experience in the field of fault diagnosis of complex digital systems allows students to inspect the taught subjects by individual research experiments. The proposed diagnostic environment is mainly dedicated for investigating digital systems but it may be used also for learning diagnostic problems for a wide class of electrical or even mechanical systems represented in a topological way as a network of components (or sub-functions) in a form of "black boxes". The proposed environment supports distance learning as well as a web-based computer-aided teaching. The interactive modules are focused on easy action and reaction, and learning by doing, encourage students for critical thinking, and educate students to be in the future creative engineers.

The described environment is introduced into the teaching process at the Tallinn University of Technology in Estonia in the programs of master study.

Chapter 8
CONCLUSIONS

The current thesis is focused on embedded fault diagnosis in digital circuits and addresses the following challenges: diagnostic resolution improvement and diagnostic application time optimization to cope with continuously growing complexities of digital circuits. To address the difficulties of handling the large variety of physical defects in nanoscale technologies caused by transistor ultimate scaling, a fault model free approach to fault diagnosis is proposed.

8.1 Main results

The contributions of the thesis are summarized as follows.

➢ **New methods for optimized fault diagnosis in BIST environments:**

- A new concept for bisectioning of test patterns for fault diagnosis is developed where instead of counting the test patterns as in the classical *Binary Search*, the diagnostic capability of patterns is selected as the criteria for bisectioning. The new bisection algorithm allows considerable reduction in the average length of diagnostic procedures.
- A sequential algorithm is developed to allow a trade-off between the speed of diagnosis and diagnostic resolution. Opposite to the classical approach which targets all failing patterns, in the proposed method not all failing patterns are necessarily needed to be fixed for diagnosis. The set of suspected faults is continuously updated during the diagnostic procedure, which allows to prune the search space and to additionally increase the speed of diagnosis.

- The proposed approach employs compressed fault tables to minimize the amount of memory space needed to store essential diagnostic data derived from a long sequence of pseudorandom test patterns.
- A new concept of test response processing to improve the accuracy of fault diagnosis in digital systems with BIST facilities is proposed. The concept is based on partitioning of a single signature analyzer into a set of multiple independent analyzers. Algorithms are proposed to design an optimal interface between the circuit under test and the set of signature analyzers to achieve the best diagnostic resolution for given test set without causing any increase in the area overhead.

In this way, simultaneously many techniques are combined together to reach the best synergy. Because of using the fault table, the proposed method classifies as a cause-effect approach, and because of organizing the diagnosis as a sequential search based on intermediate fault analysis during the adaptive search, the method has also elements of effect-cause strategy. The ultimate goal of the proposed method is to optimize diagnostic procedure by minimizing the number of diagnostic test sessions (queries) at the accepted diagnostic resolution. Experimental results demonstrate that the proposed method for increasing the diagnostic resolution and optimizing application time is feasible and efficient.

➢ **Novel hierarchical approach to fault diagnosis without using fault models:**

- A novel method for high-level fault diagnosis based on module-level fault dictionaries is developed. No fault models are used, and the objective of diagnosis is faulty module in a given network of modules. As modules, either library components (e.g. complex gates) of digital circuits or arbitrary subcircuits are considered. The method combines cause-effect and effect-cause strategies. The approach is scalable, and helps to cope with the growing complexities of digital circuits. No restrictions to fault or defect multiplicity inside the modules are set.
- A measure for characterizing the lower bound of block-level diagnosability of a given network is proposed which is based on topological fault dictionary that does not need fault simulation and represents only the connectivity of blocks to observable checkpoints. The measure can be used for redesign of the circuit to improve the exactness of locating the faults or faults regions in digital circuits by inserting additional observable checkpoints.
- In addition, a novel hierarchical approach is developed, which combines the high-level module-oriented and low-level defect-oriented reasoning to improve the diagnostic resolution in the case when information about possible defects in modules is available. Two methods are proposed for locating defects in candidate faulty modules: cause-effect reasoning based on defect libraries of modules, and effect-cause reasoning inside the modules.

- The high diagnostic resolution at the module-level was achieved by implementing a novel high-level effect-cause reasoning based on the concept of functional fault model (conditional SAF model). This concept is based on indirect mapping of physical defects from transistor level to module-level, so as to carry out indirect defect based fault reasoning at higher module-level.

The complexity of the proposed method compared to the flat logic level and fault model based diagnosis is reduced. The size of the high-level fault dictionary for the whole circuit depends linearly only on the number of modules to be determined as faulty or not faulty, and not on the number of possible faults or defects as traditionally. The size of the modules, however, will be the trade-off between the complexity of the high-level dictionary and the diagnostic resolution.

To perform experiments and to prove efficiency and feasibility of the fault diagnosis approaches proposed in the current thesis, a multifunctional e-learning environment with remote access for learning, getting hands-on experience, and carrying out laboratory research in developing optimized procedures for locating faults in complex electronic systems has been developed. It is a combination of a collection of software tools which simulate a system under diagnosis, emulate a pool of different strategies, methods, and algorithms of diagnostic reasoning and fault location, and allow to experiment with different embedded self-diagnosing architectures.

8.2 Future work

In the thesis the method for high-level fault diagnosis that employs no fault models has been proposed. This is a new paradigm for fault diagnosis, which as shown in the thesis has good potentials to localize physical defects not covered by existing fault models. The objective of diagnosis is to identify candidate faulty modules in a given network of modules without knowing which defects were causing the erroneous behavior of the module. Fault diagnosis by the algorithm developed in the thesis is performed accurately when a single faulty module is present at a time. No restrictions to fault or defect multiplicity inside the faulty modules are set. The method can also target multiple faults in different modules. However, in this case fault masking may sometimes occur, which can lead to the possible decrease of diagnostic accuracy.

Hence, further improvements of the proposed high-level fault diagnosis approach might increase the diagnostic accuracy of handling multiple faulty modules, and can be considered as one of the possible directions for future research. As one way to cope with fault masking in case of multiple faulty modules could be a method of ranking candidate faulty modules according to fault masking probabilities that could be taken into account, when pruning the sets of candidate faulty modules.

A related problem to the same multiple fault diagnosis issue is the question how to generate test patterns for detecting and diagnosing multiple faults. In case of multiple stuck-at fault assumption, an n-line circuit may have $3^n - 1$ faulty situations, which makes it impossible to generate tests on the basis of counting multiple faults. Therefore, it is more reasonable to attempt to verify the correctness of selected part of a circuit instead of targeting the faults themselves as objectives of testing. In this case, for such selected parts of a circuit, dedicated subsets or groups of test patterns should be generated, which would be able to identify or prove the correctness of the related parts of the circuit regardless the effects of all possible faults, which might be present in other places of the circuit. In this way, it would be possible to continuously narrow down the potential faulty area in the process of diagnosis. Investigations in such directions have already begun [102].

References

[1] **Chris A. Mack.** "Fifty Years of Moore's Law". – *Semiconductor Manufacturing, IEEE Transactions on*, Vol. 24, Issue 2, May 2011, pp. 202-207.

[2] **G. Moore.** "Cramming more components onto integrated circuit". – *Electronics*, Vol. 38, no. 8, April 1965, pp. 114-117.

[3] **M. Bohr.** "Nanotechnology goals and challenges for electronic applications”. – *IEEE Transaction on Nanotechnology*, Vol. 1, no. 1, March 2002, p. 56.

[4] **Ban Wong, Anurag Mittal, Yu Cao, Greg W. Starr.** "Nano-CMOS Circuit and Physical Design". – , December 2004.

[5] **P. G. Ryan, S. Rawat, W. K. Fuchs.** "Automated Diagnosis of VLSI Failures". – *IEEE VLSI Test Symposium*, Atlantic City, NJ, USA, 1991, pp. 187-192.

[6] **J. Ghosh-Dastidar, N. A. Touba.** "A Rapid and Scalable Diagnosis Scheme for BIST Environments with a Large Number of Scan Chains". – *VLSI Test Symposium*, 2000.

[7] **D. P. Vallett.** "IC Failure Analysis: The Importance of Test and Diagnostics". – *IEEE Design and Test of Computers*, Vol. 13, no. 3, 1998, pp. 76-82.

[8] **L.-T. Wang, C.-W. Wu, X. Wen.** *"VLSI Test Principles and Architectures".* Elsevier. 2006.

[9] **A. Miczo.** *"Digital Logic Testing and Simulation".* Wiley Interscience. 2003.

[10] **T. Williams, N. Brown.** "Defect Level as a Function of Fault Coverage". – *IEEE Trans. Comput.*, C-30(12), 1981, pp. 987-988.

[11] **M. Abramovici, M. A. Breuer, A. D. Friedman.** *"Digital systems testing and testable design".* IEEE Press. 1990.

[12] **C. Timoc, M. Buehler, T. Griswold, C. Pina, F. Scott, L. Hess.** "Logical Models of Physical Failures". – *Proc. Intn'l Test Conf.*, 1983, pp. 546-553.

[13] **S. C. Ma, P. Franco, E. J. McCluskey.** "An experimental chip to evaluate test techniques: Experimental results". – *in Proc. Int. Test. Conf.*, October 1995, pp. 663-672.

[14] **M. L. Bushnell, V. D. Agrawal.** *"Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits".* Springer Science, New York. 2000.

[15] **H. Fujiwara, S. Toida.** "The complexity of fault detection problems for combinational circuits". – *IEEE Trans. Comput.*, C-31(6), 1982, pp. 555–560.

[16] **N. K. Jha, S. K. Gupta.** *"Testing of Digital Systems".* Cambridge University Press, London. 2002.

[17] **E. B. Eichelberger, T. W. Williams.** "A logic design structure for LSI testability". – *in Proc. Des. Automat. Conf.*, June 1977, pp. 462–468.

[18] **P. H. Bardell, W. H. McAnney.** "Self-testing of multiple logic modules". – *in Proc. Int. Test Conf.*, October 1982, pp. 200-204.

[19] **C. E. Stroud.** *"A Designer's Guide to Built-In Self-Test".* Kluwer Academic, Norwell, MA. 2002.

[20] **E. J. McCluskey.** *"Logic Design Principles: With Emphasis on Testable Semiconductor Circuits".* Prentice Hall, Englewood Cliffs, NJ. 1986.

[21] **S. Mourad, Y. Zorian.** *"Principles of Testing Electronic Systems".* John Wiley & Sons, Somerset, NJ. 2000.

[22] **N. K. Jha, S. K. Gupta.** *"Testing of Digital Systems".* Cambridge University Press, Cambridge, U.K. 2003.

[23] **A. Jutman, A. Tsertov, R. Ubar.** "Calculation of LFSR Seed and Polynomial Pair for BIST Applications". – *11th IEEE Workshop on Design and Diagnostics of Electronic Systems*, April 2008, pp. 275-278.

[24] **G. Mrugalski, J. Rajski, J. Tyszer.** "Cellular Automata-Based Test Pattern Generators with Phase Shifters". – *in IEEE Trans. On CAD/ICAS*, Vol. 19, no. 8, 2000, pp. 878-893.

[25] **S. Chidambaram, D. Kagaris, D. K. Pradhan.** "A Comparative Study of CA with Phase Shifters and GLFSRs". – *in Proc. IEEE International Test Conference (ITC)*, 2005, pp. 926-935.

[26] **P. Fiser.** "Pseudo-Random Pattern Generator Design for Column-Matching BIST". – *in Proc. 10th Euromicro Conference on Digital System Design (DSD)*, 2007, pp. 657-663.

[27] **H. Wunderlich, G. Kiefer.** "Bit-Flipping BIST". – *in Proc. Int. Conf. On Computer-Aided Design*, 1996, pp. 337-343.

[28] **S. Hellebrand et al.** "Built-in Test for Circuits with Scan Based on Reseeding of Multi-Polynomial LFSR". – *IEEE Trans. On Computers*, Vol. 44, 1995, pp. 223-233.

[29] **A. A. Al-Yamani, S. Mitra, E. J. McCluskey.** "BIST reseeding with very few seeds". – *in Proc. 21st IEEE VLSI Test Symposium (VTS)*, 2003, pp. 69-74.

[30] **G. Jervan, P. Eles, Z. Peng, R. Ubar, M. Jenihhin.** "Test Time Minimization for Hybrid BIST of Core-Based Systems". – *Journal of Computer Science and Technology*, 2006, pp. 907-912.

[31] **J. Richman, K. R. Bowden.** "The Modern Fault Dictionary". – *in Proc. IEEE Int'l Test Conf.*, 1985, pp. 696-702.

[32] **I. Pomeranz, S. M. Reddy.** "On the generation of small dictionaries for fault location". – *in Proc. IEEE Int. Conf. on Comput.-Aided Des.*, 1992, pp. 272-279.

[33] **B. Chess, T. Larrabee.** "Creating Small Fault Dictionaries". – *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, no. 3, 1999, pp. 346-356.

[34] **J. A. Waicukauski, E. Lindbloom.** "Failure diagnosis of structured VLSI". – *IEEE Des. Test Comput.*, 6(4), 1989, pp. 49-60.

[35] **M. Abramovici, P. R. Menon, D. T. Miller.** "Critical path tracing: An alternative to fault simulation". – *IEEE Design Test Comput.*, 1(1), 1984, pp. 83-93.

[36] **A. Kuehlmann, D. I. Cheng, A. Srinivasan, D. P. Lapotin.** "Error diagnosis for transistor-level verification". – *in Proc. Design Automation Conf.*, 1994, pp. 218-223.

[37] **I. Pomeranz, S. M. Reddy.** "On correction of multiple design errors". – *IEEE Trans. Comput.-Aided Des.*, 14(2), 1995, pp. 255-264.

[38] **I.Bayraktaroglu, A.Orailoglu.** "Gate Level Fault Diagnosis in Scan-Based BIST". – *in Proc. DATE*, 2002, pp. 376-381.

[39] **A. B. Khang, S. Reda.** "Combinatorial group testing methods for the BIST diagnosis problem". – *in Proc. of the ASP-DAC*, 2004, pp. 113–116.

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

[41] **R. C. Aitken, V. K. Agarwal.** "A Diagnosis Method Using Pseudorandom Vectors without Intermediate Signatures". – *Proc. Int'l Conf. Computer-Aided Design (ICCAD)*, 1989, pp. 574-580.

[42] **W. H. McAnney, J. Savir.** "There Is Information in Faulty Signatures". – *Proc. Int'l Test Conf. (ITC)*, 1987, pp. 630-636.

[43] **C. E. Stroud, T. R. Damarla.** "Improving the Efficiency of Error Identification via Signature Analysis". – *in Proc. 13th IEEE VLSI Test Symp. (VTS)*, 1995, pp. 244-249.

[44] **J. Ghosh-Dastidar, D. Das, N. A. Touba.** "Fault Diagnosis in Scan-Based BIST Using Both Time and Space Information". – *in Proc. Int'l Test Conf. (ITC)*, 1999, pp. 95-102.

[45] **J. Rajski, J. Tyszer.** "Diagnosis of Scan Cells in BIST Environment". – *IEEE Trans. Computers*, Vol. 48, no. 7, 1999, pp. 724-731.

[46] **J. Savir, W. H. McAnney.** "Identification of Failing Tests with Cycling Registers". – *in Proc. Int'l Test Conf. (ITC)*, 1988, pp. 322-328.

[47] **Y. Wu, S. M. I. Adham.** "Scan-Based BIST Fault Diagnosis". – *IEEE Trans. Computer-Aided Design*, Vol. 18, no. 2, 1999, pp. 203-211.

[48] **I. Bayraktaroglu, A. Orailoglu.** "The Construction of Optimal Deterministic Partitionings in Scan-Based BIST Fault Diagnosis: Mathematical Foundations and Cost-Effective Implementations". – *IEEE Transactions on Computers*, Vol. 54, no.1, January 2005, pp. 61-75.

[49] **I. Bayraktaroglu, A. Orailoglu.** "Cost-Effective Deterministic Partitioning for Rapid Diagnosis in Scan-Based BIST". – *IEEE D&TC*, 2002, pp. 42-53.

[50] **M. Abramovici, M. A. Breuer.** "Fault Diagnosis in Synchronous Sequential Circuits Based on an Effect-Cause Analysis". – *IEEE Trans. Computers*, Vol. 21, no. 12, 1982, pp. 1165-1172.

[51] **J. M. Solana, J. A. Michell, S. Bracho.** "Elimination Algorithm: A Method for Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis". – *IEEE Proc. Computers and Digital Techniques*, Vol. 133, no. 1, 1986, pp. 31-44.

[52] **C. Liu.** "Compact Dictionaries for Fault Diagnosis". – *IEEE Trans. On Computers*, Vol. 53, no. 6, 2004.

[53] **H. Cox, J. Rajski.** "A Method of Fault Analysis for Test Generation and Fault Diagnosis". – *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 7, no. 7, 1988, pp. 813-833.

[54] **S. J. Sangwine.** "Fault Diagnosis in Combinational Digital Circuits Using a Backtrack Algorithm to Generate Fault Location Hypotheses". – *IEEE Proc. Electronic Circuits and Systems*, Vol. 135, no. 6, IEE Proc. G (Electronic Circuits and Systems), pp. 247-252.

[55] **S. Venkataraman, I. Hartanto, W. K. Fuchs.** "Dynamic Diagnosis of Sequential Circuits Based on Stuck-At Faults". – *in Proc. IEEE VLSI Test Symp.*, 1996, pp. 198-203.

[56] **S. Venkataraman, S. B. Drummonds.** "Poirot: a logic fault diagnosis tool and its application". – *in Proc. IEEE ITC*, 2000, pp. 253-262.

[57] **A. Rousset, A. Bosio, P. Girard, C. Landrault, S. Pravossoudovitch, A. Virazel.** "Derric: A tool for unified logic diagnosis". – *in Proc. ETS*, 2007, pp. 13-20.

[58] **V. Boppana, W. K. Fuchs.** "Fault Dictionary Compaction by Output Sequence Removal". – *in Proc. IEEE Int'l Conf. Computer-Aided Design*, 1994, pp. 576-579.

[59] **J. M. Acken, S. D. Millman.** "Accurate Modeling and Simulating of Bridge Faults". – *Custom Integrated Circuits Conf.*, 1991, pp. 17.4.1-17.4.4.

[60] **L. Zhuo, X. Lu, W. Qiu, W. Shi, D. M. H. Walker.** "A Circuit Level Fault Model for Resistive Opens and Bridges". – *in Proc. VLSI Test Symp.*, 2003, pp. 379-384.

[61] **S. K. Jain, V. D. Agrawal.** "Modeling and Test Generation Algorithms for MOS Circuits". – *IEEE Trans. Comput.*, Vols. C-34, no. 5, 1985, pp. 426-433.

[62] **H. K. Lee, D. S. Ha.** "An Efficient Automatic Test Pattern Generator for Stuck-Open Faults in CMOS Combinational Circuits". – *in Proc. Design Automation Conference*, 1990, pp. 660-666.

[63] **A. Kristic, K. T. Cheng.** *"Delay Fault Testing for VLSI Circuits".* Kluwer Academic Publishers. 1998.

[64] **J. P. Roth.** "Diagnosis of Automata Failures: A Calculus and a method". – *IBM J. Res. Develop.*, Vol. 10, no. 4, 1966, pp. 278-291.

[65] **R. D. Blanton, J. P. Hayes.** "On the Properties of the Input Pattern Fault Model". – *ACM Trans. Des. Automat. Electron. Syst.*, Vol. 8, no. 1, 2003, pp. 108-124.

[66] **Keller, K. B.** *"Hierarchical Pattern Faults for Describing Logic Circuit Failures". US Patent 5546408,* August 13, 1994.

[67] **R. Ubar.** "Detection of Suspected Faults in Combinational Circuits by Solving Boolean Diff. Equations". – *Automation and Remote Control*, Vol. 40, no. 11, 1980, pp. 1693-1703.

[68] **R. Ubar, S.Devadze, J. Raik, A. Jutman.** "Fast Fault Simulation for Extended Class of Faults in Scan-Path Circuits". – *in Proc. 5th IEEE Int. Symposium DELTA*, 2010, pp. 14-19.

[69] **U. Mahlstedt, J. Alt, I. Hollenbeck.** "Deterministic Test Generation for Non-Classical Faults on the Gate Level". – *in Proc. 4th Asian Test Symp.*, 1995, pp. 244-251.

[70] **S. Holst, H. -J. Wunderlich.** "Adaptive Debug and Diagnosis Without Fault Dictionaries". – *in Proc. of 13th ETS*, 2008, pp. 199-204.

[71] **T. Bartenstein, D. Heaberlin, L. M. Huisman, D. Sliwinski.** "Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm". – *in Proc. IEEE ITC*, 2001, pp. 287-296.

[72] **X. Wen, S. Kajihara, K. Miyase, Y. Yamato, K. K. Saluja, L.-T. Wang, K. Kinoshita.** "A Per-Test Fault Diagnosis Method Based on X-Fault Model". – *IEICE Trans. on Inf. and Systems*, Vols. E89-D, no. 11, 2006, pp. 2756–2765.

[73] **R. Desineni, R. Blanton.** "Diagnosis of Arbitrary Defects Using Neighbourhood Function Extraction". – *Proc. in VLSI Test Symposium*, 2005, pp. 366–373.

[74] **S. Holst, H. -J. Wunderlich.** "Adaptive Debug and Diagnosis without Fault Dictionaries". – *in Proc. 12th European Test Symposium*, 2007, pp. 7-12.

[75] **H. -J. Wunderlich, M. Elm, S. Holst.** "Debug and Diagnosis: Mastering the Life Cycle of Nano-Scale Systems on Chip". – *in Proc. 43rd Int. Conf. MIDEM*, 2007, pp. 27-36.

[76] **X. Wen, T. Miyoshi, S. Kajihara, L. -T. Wang, K. Saluja, K. Kinoshita.** "On Per-Test Fault Diagnosis Using the X-Fault Model". – *in Proc. ICCAD*, 2004, pp. 633-640.

[77] **P. Engelke, I. Polian, M. Renovell, B. Becker.** "Simulating resistive bridging and stuck-at faults". – *IEEE TRans. on CAD of Integrated Circuits and Systems*, Vol. 25, no. 10, 2006, pp. 2181-2192.

[78] **S. Venkataraman, S. B. Drummonds.** "Poirot: Applications of a Logic Fault Diagnosis Tool". – *IEEE Trans. Design & Test of Computers*, 2001, pp. 19-29.

[79] **Z. Wang, K. -H. Tsai, M. Marek-Sadowska, J. Rajski.** "An Efficient and Effective Methodology on the Multiple Fault Diagnosis". – *in Proc. Int. Test Conference*, 2003, pp. 329-338.

[80] **L. M. Huisman.** "Diagnosing Arbitrary Defects in Logic Designs Using Single Location at a Time (SLAT)". – *IEEE Trans. on CAD of IC and Systems*, Vol. 23, no. 1, 2004, pp. 91-101.

[81] **S. Mirzaeian, F. Zheng, K-T. Tim Cheng.** "RTL Error Diagnosis Using a Word-Level SAT-Solver". – *Int. Test Conference*, 2008, pp. 1-8.

[82] **A. Veneris, I. N. Hajj.** "Design Error Diagnosis and Correction via Test Vector Generation". – *IEEE Trans. CAD*, Vol. 18, no. 12, 1998, pp. 1803-1816.

[83] **C. Liu, K. Chakrabarty.** "Failing Vector Identification Based on Overlapping Intervals of Test Vectors in a Scan-BIST Environment". – *IEEE Trans on CAD of IC and Systems*, Vol. 22, no. 4, 2003, pp. 593-604.

[84] **S. Pateras.** "Embedded Diagnosis IP". – *in Proc. DATE*, 2002, pp. 242-244.

[85] **P. Wohl et al.** "Effective Diagnostics through Interval Unloads in a BIST Environment". – *in Proc. IEEE/ACM DAC*, 2002, pp. 249-254.

[86] **T. Clouqueur et al.** "Efficient Signature-Based Fault Diagnosis Using Variable Size Windows". – *in Proc. VLSI Design Conference*, 2001, pp. 387-392.

[87] **H. -J. Wunderlich.** "From Embedded Test to Embedded Diagnosis". – *in Proc. IEEE 10th European Test Conference*, 2005, pp. 22-25.

[88] **F. K. Hwang.** "A Method for Detecting All Defective Members in a Population by Group Testing". – *J. Amer. Statist. Assoc*, 1972, pp. 605-608.

[89] **D. -Z. Du, F. K. Wang.** "Combinatorial Group Testing and its Applications". – *World Scientific*, 1994.

[90] **Bar-Noy, F. Hwang, H. Kessler, S. Kutten.** "A New Competitive Algorithm for Group Testing". – *Discrete Applied Mathematics*, 1994, pp. 29-38.

[91] **D. -Z. Du et al.** "Modifications of Competitive Group Testing". – *SIAM J. on Computing*, 1994, pp. 82-96.

[92] **F. Brglez, H. Fujiwara.** "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran". – *Proc. of the International Test Conference*, 1985, pp. 785-794.

[93] ISCAS85 Combinational Benchmark Circuits in 'Bench' Format. Department of Computer Engineering, University of Illinois. [Online] [Cited: January 27, 2012.] http://courses.ece.illinois.edu/ece543/iscas85.html

[94] Turbo Tester toolset. [Online] [Cited: January 2, 2012.] http://www.pld.ttu.ee/tt

[95] **F. Brglez, D. Bryan, K. Kominski.** "Combinational Profiles of Sequential Benchmark Circuits". – *Proc. Int. Symposium on Circuits and Systems*, 1989, pp. 1929-1934.

[96] **R. Ubar, W. Kuzmicz, W. Pleskacz, J. Raik.** "Defect-Oriented Fault Simulation and Test Generation in Digital Circuits". – *in Proc. 2nd Int. Symp. on Quality of Electronic Design*, 2001, pp. 365-371.

[97] **F. Corno, M.S. Reorda, G. Squillero.** "RT-level ITC'99 benchmarks and first ATPG results". – *IEEE Design & Test of Computers*, Vol. 17, No. 3, 2000, pp. 44-53.

[98] ITC99 Benchmarks, Combinational Gate-Level Versions. CAD Group, Politecnico di Torino. [Online] [Cited: January 27, 2012.] http://www.cad.polito.it/tools/itc99.html

[99] **O. Novak, E. Gramatova, R. Ubar.** *"Handbook of Testing Electronic Systems".* Czech TU Publishing House. 2005. p. 395.

[100] BIST Analyzer and Diagnozer webpage. [Online] [Cited: January 3, 2012.] http://www.pld.ttu.ee/applets/bista.

[101] **A. Jutman, A. Tsertov, A. Tsepurov, I. Aleksejev, R. Ubar, H. -D. Wuttke.** "Teaching Digital Test with BIST Analyzer". – *in Proc. 19th EAEEIE Annual Conference*, 2008, pp. 123-128.

[102] **R. Ubar, S. Kostin, J. Raik.** "About Robustness of Test Patterns Regarding Multiple Faults". – *13th IEEE Latin American Test Workshop, Quito, Ecuador*, 2012 (accepted paper).

Curriculum Vitae

Personal Data

Name	Sergei Kostin
Date of birth	28.05.1984
Place of birth	Tartu, Estonia
Citizenship	Estonian

Contact Data

Address	Raja 15, Tallinn, 12618
Phone	+372 6202264
E-mail	skostin@ati.ttu.ee

Education

2007 - …	Ph.D. studies in Information and Communication Technology, Tallinn University of Technology (TUT)
2006 - 2007	M.Sc. in Computer and Systems Engineering, TUT
2003 - 2006	B.Sc. in Computer and Systems Engineering, TUT
2001 - 2003	Secondary Education from Maardu High school
1991 - 2001	Secondary Education from Tartu Annelinna High school

Carrier

2011 - …	Engineer at Department of Computer Engineering, TUT
2007 - 2011	Researcher at Department of Computer Engineering, TUT
2008 - 2009	R&D Engineer, Testonica Lab
2005 - 2007	R&D Engineer, ELIKO Competence Centre in Electronics-, Info- and Communication Technologies

Academic Degree

Master of Science in Engineering, Computer and Systems Engineering, TUT, "Fault Diagnosis in the BIST Environment Based on Bisection of Detected Faults"

Awards

2007 - 2010	Scholarship of Estonian Information Technology Foundation (EITSA)

Research topics

digital logic testing and diagnosis, built-in self-test technique,
embedded fault diagnosis optimization, fault model free diagnosis

Elulookirjeldus

Isikuandmed

Nimi	Sergei Kostin
Sünniaeg	28.05.1984
Sünnikoht	Tartu, Eesti
Kodakondsus	Eesti

Kontaktandmed

Aadress	Raja 15, Tallinn, 12618
Telefon	+372 6202264
E-post	skostin@ati.ttu.ee

Hariduskäik

2007 - …	doktoriõpe, info- ja kommunikatsioonitehnoloogia õppekava, Tallinna Tehnikaülikool (TTÜ)
2006 - 2007	tehnikateaduse magistri kraad, arvuti- ja süsteemitehnika õppekava, TTÜ
2003 - 2006	tehnikateaduse bakalaureuse kraad, arvuti- ja süsteemitehnika õppekava, TTÜ
2001 - 2003	keskharidus, Maardu Gümnaasium
1991 - 2001	keskharidus, Tartu Annelinna Gümnaasium

Teenistuskäik

2011 - …	insener, Arvutitehnika instituut, TTÜ
2007 - 2011	teadur, Arvutitehnika instituut, TTÜ
2008 - 2009	arendusinsener, OÜ Testonica Lab
2005 - 2007	arendusinsener, OÜ ELIKO Tehnoloogia Arenduskeskus

Teaduskraad

tehnikateaduse magistri kraad, arvuti- ja süsteemitehnika õppekava, TTÜ, "Poolitusmeetodil põhinev rikete diagnoos isetestivates süsteemides"

Teaduspreemiad

2007 - 2010 Eesti Infotehnoloogia Sihtasutuse (EITSA) stipendium

Teadustegevus

digitaal loogika testimine ja diagnoosimine, sisseehitatud isetestimine, rikete diagnoosi optimeerimine isetestivates süsteemides, rikkemudelita rikete diagnoos

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 Object-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 Bio-impedance. 2007.

34. **Madis Listak**. A Task-Oriented Design of a Biologically Inspired Underwater Robot. 2007.

35. **Elmet Orasson**. Hybrid Built-in Self-Test. Methods and Tools for Analysis and Optimization of BIST. 2007.

36. **Eduard Petlenkov**. Neural Networks Based Identification and Control of Nonlinear Systems: ANARX Model Based Approach. 2007.

37. **Toomas Kirt**. Concept Formation in Exploratory Data Analysis: Case Studies of Linguistic and Banking Data. 2007.

38. **Juhan-Peep Ernits**. Two State Space Reduction Techniques for Explicit State Model Checking. 2007.

39. **Innar Liiv**. Pattern Discovery Using Seriation and Matrix Reordering: A Unified View, Extensions and an Application to Inventory Management. 2008.

40. **Andrei Pokatilov**. Development of National Standard for Voltage Unit Based on Solid-State References. 2008.

41. **Karin Lindroos**. Mapping Social Structures by Formal Non-Linear Information Processing Methods: Case Studies of Estonian Islands Environments. 2008.

42. **Maksim Jenihhin**. Simulation-Based Hardware Verification with High-Level Decision Diagrams. 2008.

43. **Ando Saabas**. Logics for Low-Level Code and Proof-Preserving Program Transformations. 2008.

44. **Ilja Tšahhirov**. Security Protocols Analysis in the Computational Model – Dependency Flow Graphs-Based Approach. 2008.

45. **Toomas Ruuben**. Wideband Digital Beamforming in Sonar Systems. 2009.

46. **Sergei Devadze**. Fault Simulation of Digital Systems. 2009.

47. **Andrei Krivošei**. Model Based Method for Adaptive Decomposition of the Thoracic Bio-Impedance Variations into Cardiac and Respiratory Components. 2009.

48. **Vineeth Govind**. DfT-Based External Test and Diagnosis of Mesh-like Networks on Chips. 2009.

49. **Andres Kull**. Model-Based Testing of Reactive Systems. 2009.

50. **Ants Torim**. Formal Concepts in the Theory of Monotone Systems. 2009.

51. **Erika Matsak**. Discovering Logical Constructs from Estonian Children Language. 2009.

52. **Paul Annus**. Multichannel Bioimpedance Spectroscopy: Instrumentation Methods and Design Principles. 2009.

53. **Maris Tõnso**. Computer Algebra Tools for Modelling, Analysis and Synthesis for Nonlinear Control Systems. 2010.

54. **Aivo Jürgenson**. Efficient Semantics of Parallel and Serial Models of Attack Trees. 2010.

55. **Erkki Joasoon**. The Tactile Feedback Device for Multi-Touch User Interfaces. 2010.

56. **Jürgo-Sören Preden**. Enhancing Situation – Awareness Cognition and Reasoning of Ad-Hoc Network Agents. 2010.

57. **Pavel Grigorenko**. Higher-Order Attribute Semantics of Flat Languages. 2010.

58. **Anna Rannaste**. Hierarcical Test Pattern Generation and Untestability Identification Techniques for Synchronous Sequential Circuits. 2010.

59. **Sergei Strik**. Battery Charging and Full-Featured Battery Charger Integrated Circuit for Portable Applications. 2011.

60. **Rain Ottis**. A Systematic Approach to Offensive Volunteer Cyber Militia. 2011.

61. **Natalja Sleptšuk**. Investigation of the Intermediate Layer in the Metal-Silicon Carbide Contact Obtained by Diffusion Welding. 2011.

62. **Martin Jaanus**. The Interactive Learning Environment for Mobile Laboratories. 2011.

63. **Argo Kasemaa**. Analog Front End Components for Bio-Impedance Measurement: Current Source Design and Implementation. 2011.

64. **Kenneth Geers**. Strategic Cyber Security: Evaluating Nation-State Cyber Attack Mitigation Strategies. 2011.

65. **Riina Maigre**. Composition of Web Services on Large Service Models. 2011.

66. **Helena Kruus**. Optimization of Built-in Self-Test in Digital Systems. 2011.

67. **Gunnar Piho**. Archetypes Based Techniques for Development of Domains, Requirements and Sofware. 2011.

68. **Juri Gavšin**. Intrinsic Robot Safety Through Reversibility of Actions. 2011.

69. **Dmitri Mihhailov**. Hardware Implementation of Recursive Sorting Algorithms Using Tree-like Structures and HFSM Models. 2012.

70. **Anton Tšertov**. System Modeling for Processor-Centric Test Automation. 2012.