THESIS ON INFORMATICS AND SYSTEM ENGINEERING C109

# At-Speed Testing and Test Quality Evaluation for High-Performance Pipelined Systems

MAKSIM GOREV

TUT
PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Engineering
Chair of Computer Systems Test and Verification

**This dissertation was accepted for the defense of the degree of Doctor of Philosophy in Computer and Systems Engineering on October 30th, 2015.**

**Supervisors**: Prof. Peeter Ellervee, PhD
Dept. of Computer Engineering,
Tallinn University of Technology, Tallinn, Estonia
Prof. Raimund-Johannes Ubar, D. Sc.
Chair of Computer Systems Test and Verification, Dept. of Computer Engineering,
Tallinn University of Technology, Tallinn, Estonia

**Opponents**: Prof. Heinrich Theodor Vierhaus, PhD
Department of Computer Science,
Brandenburg University of Technology Cottbus, Germany

Prof. Erik Larsson, PhD
Department of Electrical and Information Technology,
Lund University, Sweden

Defense of the thesis: January 14th, 2016

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

Maksim Gorev



ISSN 1406-4731
ISBN 978-9949-23-876-7 (publication)
ISBN 978-9949-23-877-4 (PDF)

INFORMAATIKA JA SÜSTEEMITEHNIKA C109

# Töökiirusel testimine ja testi kvaliteedi hindamine kõrgjõudlus-konveierarhitektuuriga süsteemidele

MAKSIM GOREV

TTÜ
KIRJASTUS

*to my beloved parents Elena and Vladimir*

# CONTENTS

# LIST OF PUBLICATIONS

## PUBLICATIONS INCLUDED IN THE THESIS

1. Gorev, M.; Ubar, R.; Ellervee, P.; Devadze, S.; Raik, J.; Min, M. **Functional Self-Test of High-Performance Pipe-Lined Signal Processing Architectures.** *Microprocessors and Microsystems*,pages 1 - 14, 2015

   The author was in charge of developing the methodology and implemented all the necessary software components of the evaluation environment. The author planned and executed the necessary experiments. The author prepared the paper for publication.

2. Ubar, R.; Kousaar, J.; Gorev, M.; Devadze, S. **Combinational Fault Simulation in Sequential Circuits.** *Proceedings of International Symposium on Circuits and Systems.* Lisbon, Portugal, pages 24-27 May, 2015.

   The author developed the concept through the numerous discussions with the supervisor. The author ran the experiments. The author prepared a paper for publication.

3. Gorev, M.; Ubar, R.; Devadze, S. **Fault Simulation with Parallel Exact Critical Path Tracing in Multiple Core Environment.** *Proceedings of IEEE Conference on Design, Automation & Test in Europe – DATE-2015*. Grenoble, France, March, 2015.

   The author developed a mixed-level fault simulation concept. The author developed a concept to run PPECPT on multi-core systems. The author implemented the concept in the software. The author planned and run the experiments. The author prepared the paper for publications. The author presented the paper at the conference.

4. Gorev, M.; Ubar, R.; Ellervee, P.; Devadze, S.; Raik, J.; Min, M. **At-Speed Self-Testing of High-Performance Pipe-Lined Processing Architectures.** *Proceedings of the 31st Norchip Conference*, Vilnius, Lithuania, Nov. 2013.

   The author was in charge of developing the methodology and implemented all the necessary software components of the evaluation environment. The author planned and executed the necessary experiments. The author prepared the paper for publication. The author presented the paper at the conference.

5. Kruus, H.; Ubar, R.; Ellervee, P.; Gorev, M.; Pesonen, V.; Devadze, S.; Orasson, E.; Brik, M.; Min, M.; Annus, P.; Kruus, M.; Meigas, K. **A Benchmark Suite for Evaluating the Efficiency of Test Tools.** *The 13th Biennial Baltic Electronics Conference (BEC2012)*, Laulasmaa, Estonia, 2012.

The author participated in decision making process regarding strucural design of the circuits. The author implemented some of the circuits. The author participated in evaluation of testability characteristics of the circuits. The author took part in preparation of the paper for publication.

## PUBLICATIONS NOT INCLUDED IN THE THESIS

### *Test topics*

1. Kostin, Sergei; Ubar, Raimund; Magi, Gunnar; Gorev, Maksim. (2014). **Comparison of two approaches to improve functional BIST fault coverage.** *Proceedings of Baltic Electronics Conference – BEC.*, Laulasmaa, 6-8 October, 2014
2. Gorev, Maksim; Ubar, Raimund.**Pipelined execution of data-parallel algorithms.** *Conference proceedings of Baltic Electronic Conference 2014: Baltic Electronic Conference*, Laulasmaa, 6-8 October, 2014

### *Educational topics*

1. Jervan, Gert; Gorev, Maksim; Pesonen, Vadim (2012). **Teaching Embedded Systems as a part of a Computer Engineering Curricula.** *23rd EAEEIE annual conference*, Cagliari, Italy, February 26-27, 2012. , 1 - 4.
2. Gorev, M.; Pesonen, V.; Ellervee, P. (2010). **Introducing Computer Systems Related Topics in the First Study Semester.** *The 8th European Workshop on Microelectronics Education (EWME'2010)*, Darmstadt, Germany, May 2010.. , 185 - 188.
3. Pesonen, Vadim; Gorev, Maksim; Tammemae, Kalle (2010). **Learning-by-Gaming in HW/SW Codesign.** *8th European Workshop on Microelectronics Education*, May 10-12, 2010, Darmstadt, Germany.

### *Design topics*

1. Ojarand, J.; Min, M.; Annus, P.; Gorev, M.; Ellervee, P. (2014). **Optimization of Multisine Excitation for a Bioimpedance Measurement Device.** *In: 2014 IEEE International Instrumentation and Measurement Technology Conference Proceedings: 2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC 2014)*, Montevideo, Uruguay, May 12-15, 2014. Hoboken, NJ, USA: IEE Conference Publications, 829 - 832.
2. Gorev, M.; Pesonen, V.; Ellervee, P. (2012). **Implementation of Multisine Signal Generator for a Bioimpedance Measurement Device.** *The 13th Biennial Baltic Electronics Conference (BEC2012)*. TTU Press.

3. Pesonen, V.; Gorev, M.; Ellervee, P. (2012). **Multisine Signal Generation Method for a Bioimpedance Measurement.** *The 15th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2012)*, Tallinn, Estonia, April 2012. IEEE Computer Society, 111 - 114.

4. Jenihhin, Maksim; Gorev, Maksim; Pesonen, Vadim; Mihhailov, Dmitri; Ellervee, Peeter; Hinrikus, Hiie; Bachmann, Maie; Lass, Jaanus. (2011). **EEG Analyzer Prototype Based on FPGA.** *Poceedings of IEEE 7th International Symposium on Image and Signal Processing and Analysis (ISPA): IEEE 7th International Symposium on Image and Signal Processing and Analysis (ISPA 2011)*, Dubrovnik, Croatia, September 4-6, 2011. IEEE, 101 - 106.

5. Gorev, Maksim; Pesonen, Vadim; Ellervee, Peeter (2011). **Polynomial Curve Fitting by Substitution.** *The 8th Annual FPGAworld Conference*, Copenhagen, Stockholm, Munich, Sept. 2011.

6. Gorev, M.; Pesonen, V.; Mihhailov, D.; Jenihhin, M.; Ellervee, P. (2011). **FPGA-Based Implementation of EEG Analyzer.** *DATE'11 Friday Workshop on "Design Methods and Tools for FPGA-Based Acceleration of Scientific Computing"*, Grenoble, France, March 2011.

7. Gorev, M. ; Ellervee, P. (2010). **FPGA Based System for Video Compression and Transmission over Bluetooth.** *The 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'2010)*, Seattle, Washington, USA, Aug. 2010. IEEE, 367 - 370.

8. Pesonen, V.; Gorev, M.; Annus, P.; Min, M.; Ellervee, P. (2010). **Reconfigurable Data Acquisition Unit for Bioimpedance Measurements.** *In: Proceedings of the 12th Biennial Baltic Electronic Conference BEC2010: IEEE 2010 12th Biennial Baltic Electronics Conference*, October 4-6, 2010, Tallinn, Estonia. (Toim.) T. Rang, P. Ellervee, M. Min. IEEE, 257 - 260.

9. Pesonen, V.; Gorev, G.; Annus, P.; Min, M.; Ellervee, P. (2010). **Reprogrammable Data Acquisition Unit to Reduce Aliasing Effect in Bio-impedance Measurements.** *In: Proceedings of the 7th FPGAworld Conference: The 7th Annual FPGAworld Conference*, Copenhagen, Denmark, Sept. 2010. (Toim.) Lindh, L.; Mooney, V. J. III; Kalberg, D.; Pablo, S. de; Oberg, J.. Association for Computing Machinery, 29 - 34.

10. Gorev, M.; Ellervee, P. (2010). **Variable byte-length data compression algorithm.** *In: Proceedings of the 12th Biennial Baltic Electronic Conference BEC2010: IEEE 2010 12th Biennial Baltic Electronics Conference* ,October 4-6, 2010, Tallinn, Estonia. (Toim.) T. Rang, P. Ellervee, M. Min. IEEE, 353 - 356.

# ACRONYMS

| | |
|---|---|
| ATPG | Automatic Test Pattern Generation |
| ABIST | Arithmetic Built-In Self Test |
| AGM | Alternative Graph Model |
| ADC | Analog-Digital Converter |
| BIST | Built-In Self Test |
| BILBO | Built-In Logic Block Observer |
| BDD | Binary Decision Diagram |
| CEBE | Centre for Integrated Electronic Systems and Biomedical Engineering |
| CAD | Computer-Aided Design |
| CPU | Central Processing Unit |
| DFT | Design-For-Testability |
| DSP | Digital Signal Processing |
| DTL | Deterministic Test Length |
| DTG | Deterministic Test Generation |
| FPGA | Field-Programmable Gate Array |
| FS | Fault Simulation |
| FC | Fault Coverage |
| FFR | Fanout-Free Region |
| GPU | Graphics Processing Unit |
| GPGPU | General-Purpose Computing on Graphics Processing Units |
| HBL | Hybrid Built-In Self Test Test Length |
| HBC | Hybrid Built-In Self Test Cost |
| HyFBIST | Hybrid Functional Built-In Self Test |
| IC | Integrated Circuit |
| LFSR | Linear Feedback Shift Register |
| LBIST | Logic Built-In Self Test |
| LSB | Least Significant Bit |
| MEMS | Microelectromechanical System |
| MUX | Digital Multiplexer |
| MISR | Multiple-Input Signature Register |
| MSB | Most Significant Bit |
| MIMD | Multiple Instruction Multiple Data |
| PPECPT | Parallel Pattern Exact Critical Path Tracing |
| PPSFP | Parallel Pattern Single Fault Propagation |
| PECPT | Parallel Exact Critical Path Tracing |
| RTG | Random Test Generation |
| RAM | Random Access Memory |
| SSBDD | Structurally Synthesized Binary Decision Diagrams |
| SAF | Stuck-At Fault |

| | |
|---|---|
| SP | Scan Path |
| SIMD | Single Instruction Multiple Data |
| S3BDD | Shared Structurally Synthesized Binary Decision Diagram |
| TRE | Test Responce Evaluation |
| UUT | Unit Under Test |
| VHDL | VHSIC Hardware Description Language |
| VLSI | Very-Large-Scale Integration |
| XML | Extensible Markup Language |

# 1

# INTRODUCTION

Advantages in manufacturing of digital Integrated Circuits (IC), driven by Moore's law [50], stimulate growth in size and complexity of modern digital systems. This allowed packing more transistors into chips thus creating possibilities to implement more complex systems on a single chip. In addition, such exponential growth created challenges both for designers - how to handle such complex designs - and for manufacturers - how to test that such complex chips are working. The fundamental way to test the chip is to set its inputs to some known values - test vector, and observe its outputs whether they have the expected results or not. Single test vector is not usually enough to detect all the faults inside the chip. Therefore the sequence of different test vectors is applied. The physical defects inside the chip are described by the variety of fault models. In order to generate the test sequence the fault model is chosen first to obtain the number of possible faults. The generation of the test vectors is then done either randomly or in deterministic way [62].

In random approach the pseudo-random generator is used to find the sequence of random vectors. In order to obtain the best test sequence the equation of the pseudo-random generator and its initial value are modified. The resulted set of random sequences is then simulated on the computer model of the circuit using fault simulation to obtain the number of possible faults covered by particular sequence. In deterministic approach the software called Automatic Test Pattern Generator (ATPG) is used to analyse the model of the circuit and calculate the necessary input values required for detection of particular fault. The results for all the faults are then combined to obtain the test sequence. The test sequence is then applied either using external testers or internally inside the chip [62].

Throughout the time the role of testing in nowadays IC design and manufacturing flows becomes more and more important[9, 56], because the growing number of transistors also increases the probability of the fault occurence. With the growth of the size and complexity of the circuits the fundamental methods had to evolve in order to guarantee the correct operation of the chip, preserving the quality and speed of test.

The challenges for testing created by the technology advancements can be demonstrated on the dark-silicon example. This term refers to the percentage of the chip that has to be shut down in order to comply with thermal constraints. Different studies show that at 8 nm technology nodes this percentage could be 50%-80% [20]. Existing test methodologies should also account on this. For exam-

ple, ATPG goal is to achieve good fault coverage with short test length. In the test situation such vectors produce higher amount of switching activity inside an IC than during its normal operation. In the case of dark-silicon thermal constraints could make use of ATPG vectors infeasible, as device could simply melt during test execution [60]. Although this fact creates new challenges, in my view it could also be used to increase an amount of test hardware and provide growth in reliability and yield of future ICs.

This thesis provides solutions in some important areas of nowadays structural test, such as built-in-self-test, at-speed test, fault simulation and benchmarks. The importance of these areas is briefly introduced next.

## 1.1 BACKGROUND AND MOTIVATION

**BENCHMARKS** The design process of a digital circuit consists of solving multiple tradeoff issues according to area and speed requirements. The amount of resource sharing defines whether the circuit will be fast or compact. One thing, which is not included into typical VLSI design flow, is how the resource sharing affects the testability parameters of the circuit, such as random fault coverage, fault simulation time and the time to run ATPG algorithms [8]. Considering such things at design stage is important, because later test routines could be running for weeks and even longer in case of big industrial designs. In this situation the effect caused to the testability characteristics by design considerations could greatly affect time-to-market in the future designs. The benchmark circuits commonly used are ISCAS85[11], ISCAS89[10] and ITC99[17] that provide combinational and sequential circuits with different parameters such as random testability and different size and dimensions. There also exist specific industrial designs such as Leon [21] and OpenRISC [1] processors for methods specifically targeting general-purpose CPUs. However to the best of my knowledge there are no benchmarks that provides different design considerations in terms of resource sharing for the circuit with the same functionality.

The goal of Chapter 2 is to present the new family of benchmark circuits targeting different resource sharing considerations for the same functionality. The benchmark is based on the industrial design for computational unit of bio-impedance analyzer. The suite allows to clearly see the dependency of different testability characteristics on the amount of shared resources.

**AT-SPEED TEST** The technology advancements impose new challenges to testing of modern chips as device geometries shrink, and deep-submicron delay defects are becoming more and more important requiring more accurate dynamic tests than before [43]. Therefore testing of chips closer to real working conditions by so-called at-speed test is becoming the must. As the size shrinks down it becomes more challenging to exactly control transistor parameters [59] that results in bigger number of delay faults and lower yield. Delay faults can typically be seen, when IC works at it's normal speed and have poor detection rate at low test speeds. This issue demands new methods that target at-speed execution of

test process. The use of scan chains has proven to be often inadequate increasing the cost in terms of additional hardware and testing time [12], excessive power dissipation during test [78] and leading to yield loss because of over-testing [15].

At-speed test is an important trend today having additional benefit of the ability to test circuits under conditions that are as close as possible to normal circuit operation [29]. This factor has a direct impact on the number of chips that are found defective during system operation, but still pass all manufacturing and functional tests. At-speed testing can be used for characterization and can also expedite test application time.

A lot of research has been carried out to relieve the burden of external testers by introducing system self-test approaches like hardware-based Logic Built-in Self-Test (LBIST) that typically use Linear Feedback Shift Registers (LFSR) [62]. In LBIST, typical functions of external test equipment like test generation and response analysis are carried out on-chip, so that the tester should not handle high-speed signals externally and its role should remain only to send the test enable signals to the chip under test, and to receive the pass/fail signals. For example, scan-based and logic BIST solutions such as [51] relax the requirements on testers and considerably reduce the overall testing cost.

The question is whether a self-test sequence running in the system can adequately exercise its hardware components satisfying the targeted fault coverage requirements. Achieving the test quality target requires application of proper test sequences that is a focus in Chapter 3 of this thesis. It should also be pointed out that the quality of a test is measured not only by its fault coverage, but also by its code size (to be stored in the memory of the chip), hardware overhead, and by the test execution time.

One of the goals of Chapter 3 is to propose an approach that combines the ideas of traditional LBIST with at-speed testing to improve the test quality at less testing overhead and avoiding performance loss compared to the traditional self-test approaches. The feasibility and efficiency of the new method is demonstrated for a particular class of pipe-lined processing architectures that are easily adaptable for at-speed on-line self-testing by inherent functional sequences.

FUNCTIONAL TEST Functional testing provides a possibility to test the hardware functional paths at-speed. It is considered by engineers to test the hardware in system-like mode as thorough as possible[62]. Compared to structural testing, which is used to test individual structural elements, such as logic gates, the hardware overhead for functional testing can be smaller. This is because the inherent functionality of the system can be reused for testing purposes.

Another advantage of functional test is its ability to avoid over-testing. This term is related to testing of hard-to-detect or redundant structural faults that are rarely or not even used during operation of the chip, but require significant effort to be tested.

The disadvantage of the functional testing is comparably lower fault coverage and complexity of finding the good functional test sequence for circuits with complex structure. However More-than-Moore trend brings technologies outside

digital logic into nowadays ICs[80]. These technologies such as Micro-Electro Mechanical Systems (MEMS) or image sensors require hardware with dedicated functionality to access them in the best way possible. Dedicated hardware usually implements several simple functions to be able to run fast. This opens up a possibility to use only functional test patterns to achieve good fault coverage of such digital circuits.

One of the goals of Chapter 3 is to propose a functional test solution for a class of pipelined circuits that achieves fault coverage comparable to traditional scan-based approaches using either random or deterministic test vectors.

FAULT SIMULATION SPEED Fault simulation is one of the most important tasks in the digital circuit design and test flow. The efficiency of solving other tasks in this field like design for testability, test quality and dependability evaluation, test pattern generation and fault diagnosis relies heavily on the performance and speed of fault simulation. Such a dependence is growing especially in case of large circuits, and hence, the scalability of the fault simulation algorithms is decisive. Accelerating the fault simulation would consequently improve all the above-mentioned applications.

Fortunately nowadays advancements in multi-core systems open up different possibilities for improvement of fault simulation algorithms in terms of their performance. Available directions are pattern parallelism, fault parallelism, model parallelism and algorithm parallelism.

Algorithm parallelism stands for parallel execution of different algorithm steps. Most common is parallel execution of data read/write and computation operations. The speed-up depends on proper balance between these algorithm steps. Pattern and fault parallelism achieves good speed-up, but memory usage can be a limiting factor for scalability.

The circuit model parallelism is an interesting direction, as it allows to keep memory requirements low, at the same time providing better scalability. The reason lies in the fact that some circuit components do not depend on each other and can be simulated in parallel, just like they work in real ICs. The better scalability comes from the fact that bigger circuits have bigger number of independent gates or sub-circuits to run in parallel. The solution proposed in Chapter 4 exploits circuit parallelism for parallel fault back-tracing and the results clearly show scalability benefits.

Another fault simulation challenge lies in the area of sequential circuits. The fastest algorithms are mostly applicable for combinational circuits, while they can not be directly used with sequential circuits due to the time relations between the test vectors. One of the widely used solutions is the full-scan approach, where all the flip-flops of the circuit are made observable through the scan chain [62]. However observing every register could be expensive in terms of additional hardware and power requirements. It will be shown in Chapter 5 that there exist a way to make only a fraction of flip-flips and fan-out stems observable in order to use combinational fault simulator for simulations of sequential circuits.

## 1.2 THESIS OBJECTIVES

The goal of the thesis is to solve a series of closely related problems regarding development of BIST for high-performance pipe-lined designs. These problems target (1) the object to be tested (system under test), (2) the methods and means used for testing (BIST), and (3) the methods and means for evaluating the quality of test solutions (fault simulation).

From these problems the following main research objectives were set up in the work:

- to create a benchmark family of digital circuits with different design considerations for the same functionality.
- to develop a BIST methodology for at-speed execution using functional test patterns and targeting dedicated high-performance pipelined architectures.
- to improve the performance of parallel-pattern exact critical path backtracing for combinational circuits using general-purpose multi-core system.
- to enable combinational fault simulation of sequential circuits with reduced number of observation points.

## 1.3 THESIS CONTRIBUTIONS

This work presents the following contributions:

- the set of benchmark circuits with different amount of resource sharing and pipelined architectures is proposed for industrial design of bio-impedance analyzer. The analysis of different testability characteristics of the circuits.
- The methodology and a set of tools is developed targeting at-speed built-in self test of high-performance pipe-lined designs. The method is evaluated using bio-impedance analyzer circuit.
- The evaluation of possibility to use analog signals as a test sequence for the digital circuits .
- The implementation of the parallel exact critical path tracing algorithm for general-purpose multicore systems.
- The methodology to use a combinational fault simulation for sequential circuits with reduced number of observable points.

## 1.4 THESIS OVERVIEW

The thesis is organized as following. In Chapter 2 the family of eight benchmark circuits along with their testability characteristics is presented. The suite

consists of circuits, representing pipe-lined architectures for signal processing with the same functionality, but different structures. First the functionality and structure of the initial circuit are described. After that, structural characteristics of all benchmark circuits are presented. Then the experiments with different testability characteristics are described and results are discussed. It was shown that sharing of resources in designs may reduce the test length, but on the other hand, it will increase the time of test synthesis, and may reduce the test quality due to increase in number of fan-out reconvergencies.

Chapter 3 is dedicated to a development of a novel at-speed functional BIST methodology for the class of pipe-lined architectures of signal processors using digitized analog signals for test purposes. First an overview of the existing BIST methods is presented. After that a novel at-speed functional BIST method is described. To overcome the high cost of fault coverage evaluation of the BIST at long test sequences, a new original fault simulation environment has been developed and is discussed in the chapter. Next a case study on using digitized analog signals, as a test stimulus for the proposed BIST is presented. The experimental results regarding both the methodology and the case study are discussed in the end of the chapter.

In Chapter 4, the problems of improving the efficiency of fault simulation are discussed to provide better means for evaluating the quality of BIST in case of long test sequences. At the beginning an overview of the state-of-the-art in combinational and sequential fault simulation is presented. The overview includes both single core and multicore methods. In the next section, the theory of Parallel Pattern Exact Critical Path Tracing (PPECPT) is investigated and extended by a novel mixed-level fault simulation method. In the following, a multi-core solution for PPECPT is presented. Finally the experimental results are discussed.

Chapter 5 presents a new approach that allows using the combinational fault simulation method, developed in the previous chapter, for sequential circuits as well. A method is proposed for improving sequential circuit observability, so that the circuit could be fault simulated using any fast combinational fault simulator. The chapter starts with the short overview of available methods. After that the novel method of observability improvement is presented. Following it the experimental results are presented and discussed.

Thesis conclusions are drawn in Section 6 along with possibilities for future research.

# 2

# BENCHMARK SUITE

This chapter is based on the publication "A Benchmark Suite for Evaluating the Efficiency of Test Tools" (see Appendix A).

In this chapter a benchmark suite of eight circuits is presented and it's testability characteristics are evaluated. As the result of the cooperation in the fields of computer, electronics and biomedical engineering in the Estonian Research Excellence Centre CEBE in our laboratory we have developed a set of circuits representing different architectures of pipe-lined signal processors. The circuits have the same functionality, but they are different in structural design. This set of circuits was required for a design space exploration in order to achieve the required performance with the smallest possible resource utilization. During this research it turned out that although the functionality of the circuits was the same, the change in structure resulted in big deviation in testability characteristics of the circuits. We realized that there can be a specific relationship between different design decisions and their corresponding testability characteristics. Such a set of circuits with the same functionality, but different amount of resource sharing could be used for evaluation of the test CAD tools. In this chapter every circuit in the suite is evaluated in terms of its testability and fault analysis characteristics in comparison with circuits structural complexity. The results are also presented and discussed.

The author's contribution lies in the development of the circuit variations, their corresponding implementation and evaluation of testability characteristics.

The rest of the chapter is organized as following. The overview of the initial design for benchmark suite and it's functionality is presented in Section 2.1. Section 2.2 discusses structural changes in the circuits forming the suite. The experimental study on the different testability characteristics of the proposed benchmark circuits is presented in Section 2.3. Section 2.3.1 provides the discussion about these characteristics. Finally the conclusions of the chapter are drawn in Section 2.4.

## 2.1 OVERVIEW OF THE INITIAL DESIGN

As a basis for the suite a bio-impedance signal analyzer circuit that implements a simplified signal processing algorithm has been chosen [5, 19]. This design was selected because of its pipe-lined structure, where every stage has a parallel implementation that can be joined to produce certain level of reconvergency.

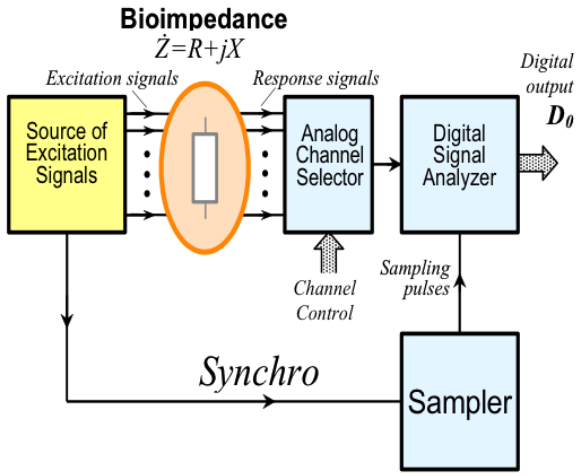



**Figure 2.1** – The architecture of digital multichannel bio-impedance analyzer (DMBA).

The device is dedicated to bio-impedance measurement of biological tissue. The synchronized excitation of sine wave of different frequencies is applied through different channels to the tissue at one point and measured as a voltage response at the other point of the tissue. A typical digital solution is that the response voltage is digitized in an analog-to-digital converter (ADC) into a uniformly sampled train of digital data that is then processed numerically in a digital signal processing (DSP) unit often using the Discrete Fourier Transform (DFT). Because the whole signal path from the generation of the set of excitation signals to the analog-to-digital conversion procedure and data analysis is synchronous by design, optimized signal processing methods can be applied. Using of sampling, which is synchronous to the known excitation waveform, enables to use a simplified, but much faster signal processing than the Fourier Transformation is. When sampling the response signal uniformly with intervals t = T/4, where T is a period of the signal, the following simple mathematics is valid [19, 46].

(1) the direct current component DC can be determined as

$$\begin{aligned} DC &= (Re_+ + Re_-)/2 \\ DC &= (Im_+ + Im_-)/2 \end{aligned} \tag{2.1}$$

(2) the real Re and imaginary Im parts of the phasor of complex bio-impedance Z is determined as

$$\begin{aligned} Re &= (Re_+ - Re_-)/2 \\ Im &= (Im_+ - Im_-)/2 \end{aligned} \tag{2.2}$$

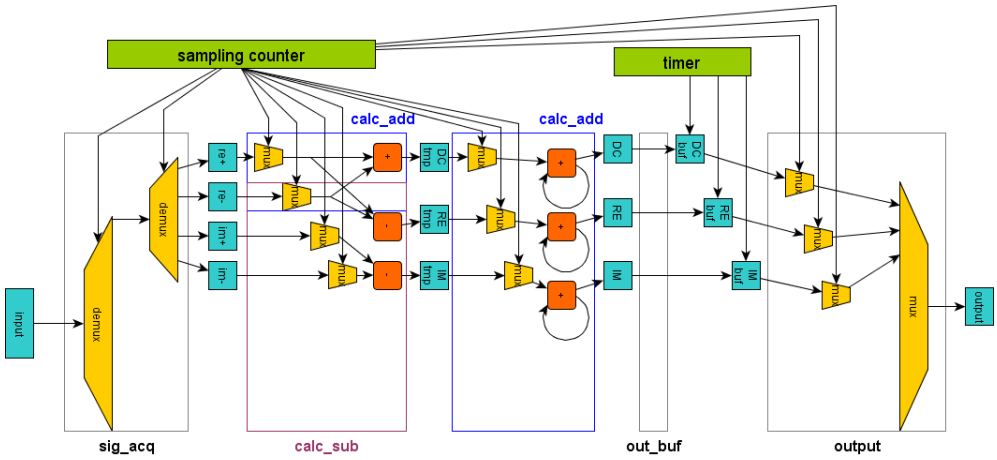



**Figure 2.2** – A block diagram of one channel of DMBA [19]. Registers are blue, MUXes – yellow, computation - red, control – green.

The mentioned signal analyzer is a part of the developed digital multichannel bio-impedance analyzer (DMBA) [5], depicted in Fig. 2.1. The Source of Excitation Signals generates digital waveforms that are converted by Digital Analog Converter (DAC, not shown) into analog signals, sent through tissue, collected by Analog Channel Selector and converted by ADC (not shown) back to digital form. The Sampler is used to synchronize the signal source and ADC (input of the Digital Analyzer, not shown). This sampled digital signal is processed by the analyser unit.

Fig. 2.2 shows the architecture of the analyzer unit. As it can be seen, the circuit has a pipe-lined structure in order to be implemented in a low-cost FPGA. The signal is first sampled into the input buffer. On the next stage signal points are distributed to four registers defining different signal components Re+, Re-, Im+ and Im-. These four registers are present in all 8 different channels of the analyzer. The sampling is performed on channel-after-channel basis. Every sample out of 4 samples taken per channel is saved into its corresponding 16-bit register of particular signal component. On the next stage Real, Imaginary and Direct current components are computed with adders and subtractors, using equations (2.1) and (2.2). On the next pipeline stage the computed components are integrated using adders and saved into 32-bit registers, called output buffers. Integration is made over a 1 ms period. After that the values of the output buffers are transferred to the output register of the analyzer.

The architecture of the analyzer was defined by the used technology - low-cost FPGA-s - and required 80 MHz sampling frequency. This constraint came from the need to have 10-20 MHz excitation signals with 4 or 8 sampling points per period [5]. The use of single input channel (ADC output) and sorter to reduce aliasing effect [19] resulted in two first buffer stages. The two last buffer stages

are defined by the need to accumulate the collected data over 1 ms period, buffer it and transmit for further analysis [5]. The intermediate part - subtractor/adder and accumulator - can be implemented, in principle, as a single stage. However, when using FPGA-s, the extra pipeline stage actually makes the design not only faster (because of the shorter combinational paths), but also smaller - every output bit of an adder has a flip-flop anyway and the use of them makes routing problem for the design tools easier. The same applies for the potential reuse of functional units and registers that would essentially add additional multiplexors to the design - the internal structure of FPGA-s is best suited for pipelined data-stream oriented applications. This was also the case with the other implementations of the same processing unit with the different degree of reuse [19]. All the MUXes in the circuit are used for switching the channels (there are 8 channels in the DMBA), except the last one that is switching the calculation results of every channel to a single output.

## 2.2 STRUCTURAL CHARACTERISTICS OF THE CIRCUITS

One of the strong parts of the suite developed is the fact that different circuits represent different design alternatives that were considered during actual design process. The suite gives a possibility to study how the different design approaches with the same functionality would impact testability - that is not often considered at the design stage. Seven implementations of the original highly parallel design were developed in a way circuit designer would consider doing. These modifications bring different degree of reconvergency to different parts of the circuit by sharing some of the components. The whole analyzer can be sub-divided into several functional parts: acquisition, preprocessing, integration, output. During the design process, alterations were made to first three parts of the analyzer: acquisition, preprocessor and integrator.

The idea of the benchmark circuits is to selectively share some of the resources in order to produce different circuit implementations of the same functionality. This resulted in eight different configurations performing the same function: 8*a*, 8*b*, 8*be*, 8*bk*, 8*bs*, 8*c*, 8*d* and 8*de*. Fig. 2.3 shows which successive changes were introduced into the designs. The difference of circuits 8*a*, 8*b*, 8*be*, 8*bk* and 8*bs* from 8*c*, 8*d* and 8*de* lies in additional 4 extra buffer registers in signal acquisition part of the circuit. All the other modifications are regarding sharing of the preprocessor and integrator resources in different manner. Table 2.1 shows which parts of the design were modified in every implementation. This table presents the number of adders and subtractors, number of bits in registers and area in equivalent gates, calculated by Synopsys Design Compiler.

Here the design 8*a* is the initial version with 8 data channels, described in Section 2.1. The transition from 8*a* to 8*b* lays in replacement of 8 channels in preprocessing part of the circuit by a single common channel, thus removing the redundancy by a shared preprocessor. It explains the decrease in adders, subtractors and registers located in preprocessor. In this version of the circuit the one single preprocessor block is shared among 8 channels of the analyzer in channel-

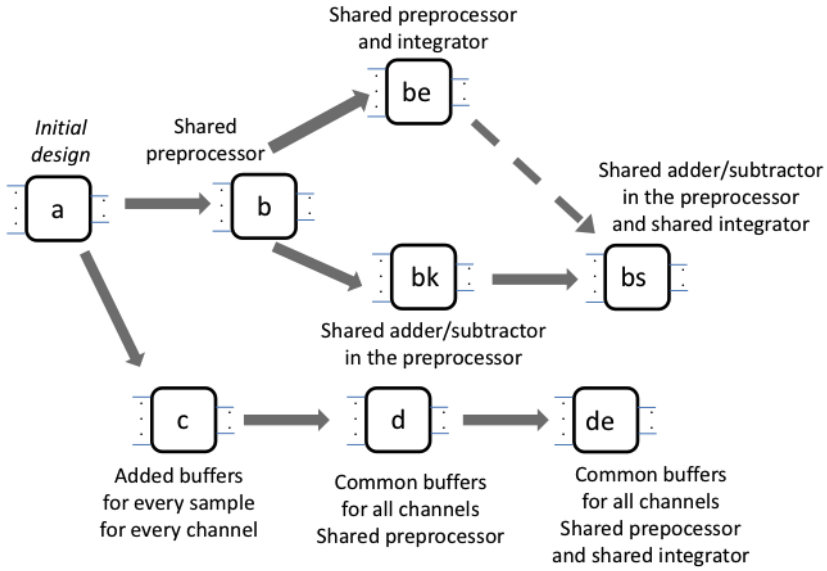



**Figure 2.3** – Overview of the benchmark designs

after-channel manner. This resulted in multiple multiplexors in the circuit that increased reconvergency in the beginning of it. Table 2.2 shows estimation of the number of reconvergent signals for each benchmark.

The transition from 8*a* to 8*c* resulted in implementing the extra input buffers. The increase in register bits can also be seen from the Table 2.1. The 8 channels of

**Table 2.1** – Structural differences of 8 benchmark circuits.

| Circuit | Structure | | | | Area, ge | | |
|---|---|---|---|---|---|---|---|
| | # 16-bit adder | # 32-bit adder | # 16-bit subtract. | Registers, bits | logic | registers | total |
| 8a | 8 | 24 | 16 | 2489 | 26182 | 17577 | 43759 |
| 8b | 1 | 24 | 2 | 1705 | 20232 | 12089 | 32321 |
| 8be | 1 | 3 | 2 | 1705 | 22090 | 12089 | 34179 |
| 8bk | 1 | 24 | 1 | 1705 | 20088 | 12089 | 32177 |
| 8bs | 1 | 1 | 1 | 1705 | 22671 | 12089 | 34760 |
| 8c | 8 | 24 | 16 | 3004 | 28177 | 21182 | 49359 |
| 8d | 1 | 24 | 2 | 2668 | 28136 | 18830 | 46966 |
| 8de | 1 | 3 | 2 | 2668 | 29850 | 18830 | 48680 |

**Table 2.2** – Difference in level of reconvergency of 8 benchmark circuits.

| Design | # of reconvergent signals | | | Modifications |
|---|---|---|---|---|
| | Preprocessor | Integrator | Total | |
| 8a | 32 | 64 | 96 | Initial design |
| 8b | 32 | 64 | 96 | Shared preprocessor |
| 8c | 32 | 64 | 96 | Initial design with extra input buffers |
| 8d | 32 | 64 | 96 | Extra input buffers and shared preprocessor |
| 8bk | 64 | 64 | 128 | Shared preprocessor with single shared adder/subtractor |
| 8be | 32 | 512 | 544 | Shared preprocessor and integrator |
| 8de | 32 | 512 | 544 | Extra input buffers, shared preprocessor and integrator |
| 8bs | 64 | 1536 | 1600 | Maximum sharing of resources |

data remained for preprocessor and integrator. That is why the number of adders and subtractors is the same for both circuits. Extra registers brought additional latency to data path, but didn't add any reconvergency.

The circuit 8*d* has union of changes in 8*b* and 8*c* - the extra registers are added to the input buffers and preprocessor is shared between all the channels that brought the amount of adders and subtractors down.

The bigger changes in reconvergency start to be seen in circuit 8*bk*. Additionally to single preprocessor block it also shares the inner part of the preprocessor. It calculates three different characteristics of the signal using single adder and subtractor unit for this purpose instead of three in all previously described circuits. This change added additional multiplexors to the preprocessor part that can be seen from Table 2.2. The number of reconvergent signals rose from 32 to 64.

Apart from 8*bk*, circuit 8*be* still has separate adder and subtractor blocks inside preprocessor. However in addition to the modifications of 8*b* it also has one single integrator block shared with all 8 channels. This led directly to the decrease in 32-bit adders located inside an integrator. The multiplexors were added to the inputs of 32-bit adders in order to enable this functionality.

The circuit 8*de* is an analog of 8*be*. The only difference lies in the addition of extra buffers to the inputs that can be seen in increased number of register bits. It doesn't change the amount of reconvergency, but increased the latency of the pipeline.

The circuit 8*bs* combined all the possible reconvergencies by sharing single preprocessor block for all the channels. This block contains single adder and subtractor unit for calculation of all three characteristics of the signal. And finally the single integrator block is shared among all the 8 channels. These changes resulted in the most compact circuit with only 3 computational units and a lot of multiplexing. It can be seen from Table 2.2 that the number of reconvergent signals increased considerably to the total of 1600 for this circuit.

## 2.3 TESTABILITY CHARACTERISTICS OF THE CIRCUITS

The goal of the testability evaluation was to investigate how different level of reconvergency and its location inside the circuit would impact the testability and fault analysis. The changes in design alternatives are characterized by different structural complexities that will have a direct impact on testability of circuits and on the testing quality. The experimental results presented in Table 2.3 and Fig. 2.4 allow to easily create functional dependencies between the testability features and the resource sharing options in design alternatives that allows to find proper tradeoffs. In the following the depicted results are discussed in details .

Testability analysis of different configurations of the bio-signal processing design was performed by using deterministic and pseudorandom test pattern generators [57], fault simulator [70] and by using the algorithms for hybrid BIST optimization developed in [38]. Several testability characteristics presented in Table 2.3 were analyzed: the deterministic test length achieved (DTL) and the needed time for deterministic test pattern generation (DTG), the time needed for fault simulation (FS) and for the pseudorandom test simulation (RTS), the hybrid BIST length (HBL) and the calculated optimal test cost of hybrid BIST (HBC). Generation times are given in seconds, test lengths in numbers of patterns, and costs in abstract units. The changes in testability characteristics for the benchmark suite are shown in Fig. 2.4.

### 2.3.1 *Discussion of the results*

The changes in design alternatives are characterized by different structural complexities that will have a direct impact on testability of circuits and on the testing quality. Here the experimental results presented in Table 2.3 and Fig. 2.4 will be discussed in details.

The transition from 8*a* to 8*b* resulted in improvement of all the testability characteristics. The best improvements were in reduction of test synthesis time (for deterministic test 1.4 and for pseudorandom test 1.25 times). Fault simulation became 1.16 times faster. The cost of the hybrid BIST significantly improved – one

**Table 2.3** – Testability characteristics of signal processors.

| Design | DTL | DTG | FS | RTS | HBL | HBC |
|---|---|---|---|---|---|---|
| 8a | 1364 | 47 | 13.7 | 1408 | 23 038 | 197 823 |
| 8b | 1201 | 34 | 11.8 | 1130 | 18 540 | 138 324 |
| 8bk | 1288 | 35 | 11.3 | 1129 | 17 497 | 144 876 |
| 8c | 1320 | 75 | 15.5 | 1583 | 35 641 | 224 121 |
| 8d | 1394 | 62 | 16.6 | 1647 | 32 610 | 209 384 |
| 8be | 995 | 114 | 27.9 | 2784 | 14 202 | 104 474 |
| 8de | 1096 | 112 | 33.4 | 3344 | 33 968 | 162 557 |
| 8bs | 1186 | 296 | 69.0 | 7095 | 14 086 | 113 038 |

of the reasons is smaller number of inputs in 8*b* that results in the less cost of the memory component of the BIST.

In transition from 8*b* to 8*be* the deterministic test length improved - it was 1.2 times shorter that can be explained by the reduction of the circuit complexity. On the other hand, the time needed for deterministic test generation was 3.35 times higher because of the increased number of reconverging signals in the circuit that causes higher number of backtracks during search for consistent solutions. Also, fault simulation time became 2.36 times slower, and the time needed for pseudorandom test simulation was 2.46 times higher. This is explained by the use of exact critical path tracing algorithm [70] used for fault simulation that is highly sensitive to the number of reconvergent fan-out stems. The cost of the hybrid BIST was improved due to the smaller number of deterministic vectors needed.

During the transition from 8*b* to 8*bk* the increase of reconvergency (from 96 to 128) did not significantly influenced the testability of the circuit.

Opposite in transition from 8*bk* to 8*bs* the number of reconvergent signals increased drastically (128 for 8*bk* and 1600 for 8*bs*). Fig. 2.4 shows worsening of the testability regarding test generation and fault simulation: the time of deterministic test generation became 8.45 times longer, the time of fault simulation 6.1 times longer and the time of random test simulation became 6.28 times longer. On the other hand, because of the reduction in circuit size, the length of deterministic test set became slightly shorter (1.08 times). The length of optimal hybrid BIST was 1.24 times shorter and optimal cost 1.28 times smaller due to the smaller number of seeds for LFSR.

In Fig. 2.4 it can be seen that due to transition from 8*a* to 8*c* the time related characteristics have become worse: the generation time for deterministic test became 1.59 times longer. The fault simulation became 1.13 times slower for both deterministic and random test patterns. This worsening of indicators can be explained by the increase of the number of reconvergencies because of adding

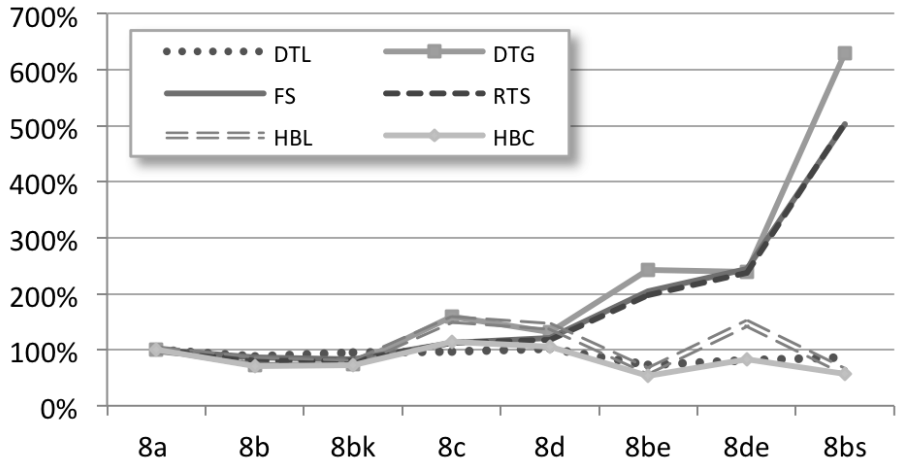



**Figure 2.4** – Changes in testability characteristics.

control signals for addressing the buffer registers. The test length did not change because the circuit size remained the same. The length of hybrid BIST sequence test became 1.54 times longer, and the cost of Hybrid BIST was bigger for 8*c* due to the bigger number of inputs (buffer registers).

In the transition from 8*c* to 8*d* the characteristics that changed most significantly were deterministic test generation time (became shorter) and the length of the optimal hybrid BIST became slightly shorter, similarly as in the case of "from 8*a* to 8*b*".

In transition from 8*d* to 8*de* caused the increase in number of reconvergent signals (Fig.2.4), and longer times for test generation and fault simulation. Deterministic test generation took 1.81 times longer. Fault simulation became 2.01 times slower for determinsic test and 2.03 times longer for random. The deterministic test set was 1.27 times shorter and the cost of the BIST reduced 1.28 times (due to the smaller number of seeds). This case affected the testability characteristics in the similar way as in the case from "8*b* to 8*bk*".

## 2.4 CHAPTER SUMMARY

In this chapter the following main results were achieved:

- A novel suite of benchmark circuits was developed and investigated to give a possibility for systematic characterization of CAD tools by creating functional dependencies for different testability markers on the structural complexity in terms of the number and configuration of reconvergencies in circuits. Existing benchmark suites do not provide such a possibility.
- It was shown that sharing of resources in designs may reduce the test length, but on the other hand, it will increase the time of the test synthesis,

and may reduce the test quality due to increasing of number of fan-out reconvergencies.

As the result of the cooperation in the fields of computer, electronics and biomedical engineering in the Estonian Research Excellence Centre CEBE, a benchmark suite was developed for evaluating the CAD tools in their efficiency and quality in designing dependable digital systems.

Differently from all other existing benchmark suites, all the member processors of this family perform the same function, but are implemented in different ways, differing mainly in the amount of shared computing resources. This gives an excellent possibility for direct systematic characterization of CAD tools by creating functional dependences for different testability markers on the structural complexity of circuits.

The experiments show a correlation between the structural properties of circuits and their testability characteristics. It was shown that sharing of resources in designs, which leads to increasing number of fan-out reconvergencies, may reduce the test length, but on the other hand, will increase the time of test synthesis, and may reduce the test quality.

A useful synergy was achieved by creating a selection of bio-signal processors that will have practical use in medical field, but simultaneously can be used as a family of benchmark circuits for analyzing the properties of new test algorithms in the field of electronics.

The new at-speed BIST methodology described in the next chapter was evaluated using benchmark circuits described here.

# 3

# AT-SPEED SELF-TESTING

This chapter is based on the publications "Functional Self-Test of High-Performance Pipe-Lined Signal Processing Architectures" (see Appendix C) and "At-Speed Self-Testing of High-Performance Pipe-Lined Processing Architectures" (see Appendix B).

In this chapter, the new approach for self-testing of digital systems with pipelined architectures is proposed. The main contributions of the chapter are the following:

- The new at-speed functional BIST methodology for high-performance pipelined architectures.
- Novel evaluation environment to transfer sequential fault simulation task into a set of combinational subtasks is developed.
- Exploration of the potential of digitized analog signals to be used as a test-sequences for at-speed BIST.

The at-speed BIST methodology forms the core of the chapter. It exploits inherent functionality of the system to produce internal test-sequences, thus requiring minimal hardware overhead for testing purposes. The method targets at-speed execution and uses single stuck-at fault model. The evaluation environment is proposed to improve the speed scalability of the method. Also digitized analog signals are explored to be used along with the proposed at-speed BIST methodology to reduce hardware requirements.

The author is in charge of the development and evaluation of the methodology proposed, running the experiments and analysis of the results.

The rest of the chapter is organized as follows. Section 3.1 presents and overview of the state-of-the-art. The general description of the novel at-speed BIST methodology is written in section 3.2. The bio-impedance analyzer from the benchmarks proposed in chapter 3 has been used in order to describe and evaluate the method. The section 3.3 describes the fault simulation environment intended to be used with the methodology proposed. Later in section 3.4 this methodology is evaluated and study the potential of different digitized analog sequences to be used as a test sequence for functional BIST. The experimental results are discussed in Section 3.5 and finally draw a conclusion in section 3.6.

## 3.1 OVERVIEW

Built-in self-test (BIST) uses on-chip hardware to both generate the test patterns and analyze an output response of the Unit Under Test (UUT). The common way to do this is to use pseudo-random pattern generator for generation of a test and multiple-input signature register (MISR) to make a signature from circuit responses. Once the self-test is done the signature is shifted out or compared against predefined value on chip. In case of incorrect signature the chip fails the test [62].

Throughout the years of research many variations of BIST were developed. In traditional Logic BIST (LBIST), test pattern generation is mostly performed by Linear Feedback Shift Registers (LFSR) [62], cellular automata [29] or multifunctional registers like Built-in Logic Block Observer (BILBO) [62] to apply pseudorandom patterns to the Unit Under Test (UUT) and to analyze its output responses. However, many circuits contain random-pattern-resistant faults that limit the fault coverage that can be achieved with this approach.

One method to improve the fault coverage for LBIST is to modify the UUT by either inserting test points [15] or by redesigning it to improve the fault coverage [28]. The drawback of these techniques is that they generally add additional logic levels to the circuitry that can degrade system performance. Another possibility to improve the fault coverage is to use weighted pseudorandom test sequences [3]. The disadvantage of this approach is in the need of storing of the weight sets on chip, and also dedicated control logic is required to switch between weights, so the hardware overhead may become large. A "mixed mode" approach, where deterministic patterns will be added to detect hard-to-test faults, has been developed in [27, 31, 35, 64]. In [35] a technique based on reseeding LFSR was proposed that reduces the storage requirements. In [27], multi-polynomial LFSR for encoding a set of deterministic test cubes was introduced, and in [31] a technique called bit flipping for generating deterministic test cubes using BIST control logic was proposed. Further, in [64] a mixed-mode approach was presented in which deterministic test cubes are embedded in the pseudorandom sequence of bits itself.

As it have already been mentioned earlier the established BIST solutions use special hardware (typically LFSR) for test pattern generation (TPG) and test response evaluation (TRE) on chip [62], but this in general introduces significant area overhead and performance degradation. To overcome these problems, specialized methods were proposed, which exploit specific functional units such as arithmetic units for on-chip test pattern generation [18, 76] that may afford to reach similar fault coverage like traditional LFSR-s. These methods are called Arithmetic BIST (ABIST), since they essentially adopt the additive congruential generation scheme of pseudo-random numbers [33].

In [32, 73], a mixed-mode or hybrid BIST approach was proposed, where a test set is assembled from two parts, from pseudorandom test patterns that are generated on-line, and deterministic test patterns that are generated off-line and stored in the system. A combination of both test sources in an optimized fashion

allowed improving the traditional LBIST in targeting hard-to-test faults. A similar approach called Hybrid Functional BIST (HyFBIST), where instead of LBIST the inherent functional sequences were used, was proposed in [44, 72] for testing digital systems, and particularly micro-programmed data-paths.

In this chapter I generalize and combine the ideas of using inherent functional blocks for test generation [18, 58] and the inherent working sequences produced by the UUT itself for self-testing purposes. An overall functional self-test concept is proposed for pipelined architectures where the working sequences are produced on the primary inputs of the system and Multiple Input Signature Analyzers (MISR) monitor the internal signals in selected test-points. A systematic procedure is proposed for selecting the test-points to achieve the best overall fault coverage at minimum testing overhead and cost.

To my knowledge, the usage of digital representation of analog signal sequences as a functional test for testing digital circuits (signal processing architectures) is investigated in the first time. Main idea is to take the input data, which is close to what the circuit-under-test would most probably have during its normal operation, and apply this data as an at-speed test. In the case of this work the input data is a digital representation of the sine signal. It will be shown in results that such a signal could yield better fault coverage in comparison to traditional pseudo-random LFSR sequence. This can also be considered as one step further compared to the arithmetic BIST (ABIST), since the source for the first stage of UUT is stimulated using more complicated equation (sine wave), than traditionally used in ABIST. The next stages of the UUT can be considered as test generators similar to ABIST. The Functional test strategies (e.g. software based self-test) used for example in microprocessors, are traditionally using dedicated software test routines that have to be stored in the memory. In case of proposed method there is no need to store in the memory such test routines or other test data.

## 3.2 GENERAL DESCRIPTION OF THE METHOD

Consider a digital system as a network of sub-circuits (blocks) where all the blocks may play simultaneously two roles: on one hand, each block will be itself UUT, and on the other hand, it will serve as the test pattern generator for the subsequent blocks it is feeding. As the overall test source, selected input working sequences (as functional test) will be used.

Two main problems arise: (1) how to find the best functional test sequences, and (2) how to find the minimal set of test-points for monitoring to achieve the highest fault coverage of testing.

In some cases, the first problem can be solved straightforwardly like in the instruction set architectures or in signal processing units. In the first case, the instructions can be exercised one by one where the problem recedes to finding only proper data (operands) as test patterns [22, 71]. In case of signal processing units, the analog signals to be processed can be used as candidates for exploiting in testing purposes as well. The possibility of using given digital representation

of analog signals as stimuli for testing signal processors is therefore also investigated in this chapter. The case study on this topic is presented in Section 3.4. The idea is similar to random (LFSR based) testing where the critical point is analysis of the test quality as the function of test length.

For example, in bio-impedance spectroscopy, for measuring the bio-impedance typically the following signals are generated and processed as shown in Fig.3.1: sine [45] and chirp [52]. These signal sequences may be used as well in the role of stimuli (i.e., functional test sequences) for self-testing purposes for the same signal processor itself. The quality of the listed signals as test stimuli can be com-

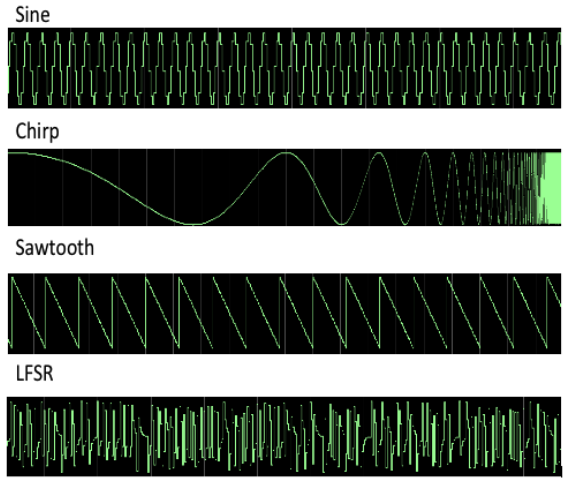



**Figure 3.1** – Signals used for measuring bio-impedance

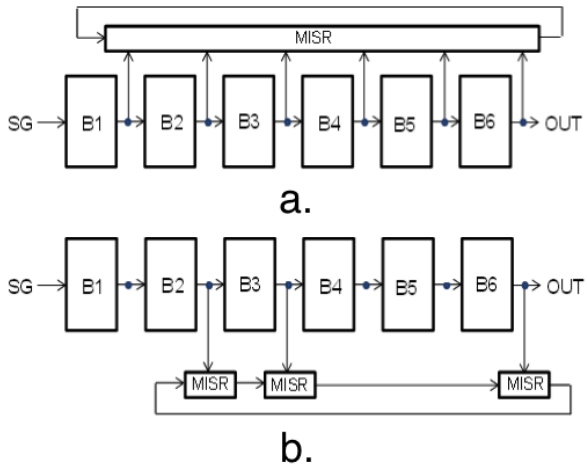



**Figure 3.2** – Monitoring of the pipe-lined signal processing unit. a. Full MISR coverage, b. Partial MISR coverage

pared with popular saw-tooth analog signal and pseudorandom LFSR sequences that are traditionally used in the logic BIST solutions. Saw-tooth is easy to generate digitally; this is the reason why it is widely used in signal generation and processing. It can also be thought of as an additive generator of exhaustive patterns. The impact of presented signals on the testability of the pipe-lined circuit is described in Section 3.4.

The second problem of selecting test-points for monitoring the test process depends how well can the faults in different blocks be detected by the given functional test sequence.

In Fig.3.2, an example of a pipe-lined signal processing unit is given that is partitioned into 6 blocks. Two solutions are demonstrated for monitoring the behavior of the circuit with MISRs. The solution in Fig.3.2a shows the case where all blocks are monitored whereas in the solution depicted in Fig.3.2b, only three MISR are used: the first is monitoring the behavior of blocks B1 and B2 as a whole, the second MISR is monitoring solely the block B3, and the third MISR is monitoring the blocks B4, B5 and B6 as a whole.

The task of partitioning of the whole system into UUT blocks has the goal to find the highest fault detection coverage for the given functional test by achieving well-balanced testability at the minimum number of monitoring points equipped with MISR. Optimal partitioning is rarely possible without any feedback on the testability of the circuit. It means that in general case a number of iterations is required in order to find the right balance between the required level of testability and number of monitors (area overhead). The general iterative procedure is described next in order to find the optimal solution.

To find the minimum hardware overhead I propose the following method for selecting test-points:

- Put MISR on the primary output of the circuit, and find the fault coverage (FC) for the given test sequence.
- If FC is sufficient, then the problem is solved.
- Partition the circuit into a set of n blocks (each with its own MISR). Find FC for each block as a UUT.
- Continue the partitioning of the blocks with low FC until the total FC will be sufficient.
- Integrate the consecutive blocks with high FCs into UUTs (with a single joint MISR in the output of the composite block) to minimize the number of MISR, so that the total FC of the system remains sufficiently high.

The described method is illustrated in Fig.3.3. Please note that the partitioning solutions can be found in different ways, e.g. dictated by an inherent structure (network of registers and combinational blocks), using any ad-hoc method in a style of "trials and error" or using more sophisticated analysis methods. This task should be regarded as a separate problem, not discussed here.

Let's consider an example shown in Fig.3.4. Here initially the circuit is partitioned into three blocks for monitoring. The fault simulation of this configuration

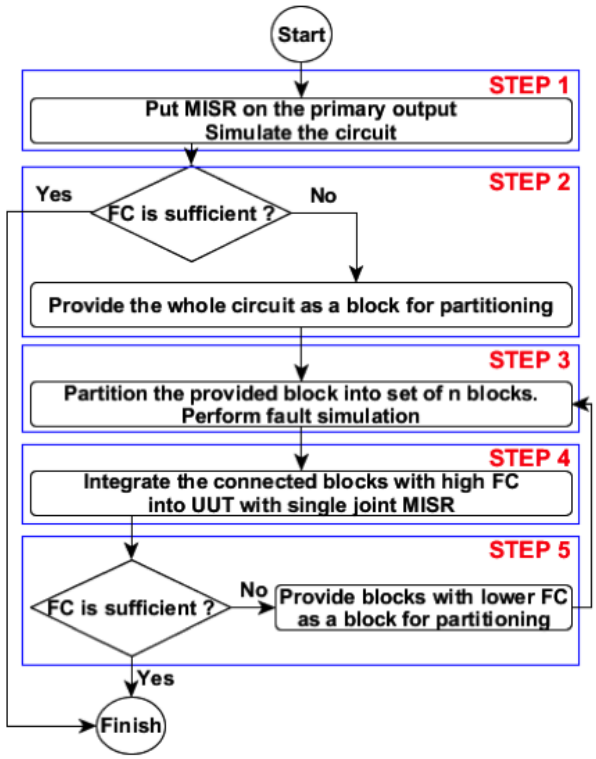



**Figure 3.3** – General procedure for minimization of the number of observation points.

shows that the fault coverage of the first two blocks is good so they can be merged in order to remove the monitors after the first block. The only MISR registers to observe the functionality of this now combined block would be the ones placed after the second block. On the other hand the third block shows low fault coverage, which suggests that the amount of observation points should be increased inside the block. This is why this block is partitioned into three sub-blocks and monitors are added after every sub-block in order to improve the fault coverage. The bottom part of the Fig.3.4 shows that first two blocks from the top picture now became one block. Also the last block now becomes divided into three blocks with MISRs inserted after each of them. The evaluation of the fault coverage should be carried out again separately for all the four parts, to find out if some of them can be merged, or if there is any of them with low fault coverage that should be further partitioned.

In case when the fault coverage will not satisfy either globally for the whole circuit or for particular blocks as UUTs, either the better functional test sequences should be found, or different methods, similar to the ones for improving LBIST described in Section 3.1, may be used. Another possibility is to use ad-hoc testability improvement in terms of control points. Control points can be used with at-speed test execution by applying them for the whole period of a test sequence.

This way the hardware overhead can be kept low and at-speed execution is still guaranteed. However investigation of these possibilities is out of the scope of current work and can be considered as a future research in this direction.

The described method of inserting MISR facilitates the idea of LBIST strategy. The only exception is the use of inherent test functionality of existing hardware in the system, instead of dedicated test generators of LBIST. This way the method affords at-speed testing with no performance degradation and with little hardware overhead and reduced test cost. The clock cycle based observation technique allows to avoid fault masking, and to achieve high fault coverage. The test response observation is carried out using built-in MISR as the only hardware overhead.

In summary the proposed method has several advantages compared to the state-of-the-art scan-path based LBIST methods:

- no hardware test pattern generators, and no scan-path for shifting in external test patterns are needed that results in smaller overhead;
- compared to LBIST, the typical drawback of over-testing related to LBIST is avoided, since only functional working test patterns are used;
- testing is carried out in the normal working clock-rate that guarantees at-speed exercising of the whole circuit.

The target of this Section was to describe the main principles of redesign for better testability of the given UUT. The goal was not to develop exact algorithm or tool for exploring automatically the whole space of solutions that would be infeasible in general case. The designer has a possibility to remove or insert MISRs in the design and to evaluate the test quality by using the fault simulation environment described next in Section 3.3. He has also the possibility of changing the length of the test sequence to achieve higher fault coverage.

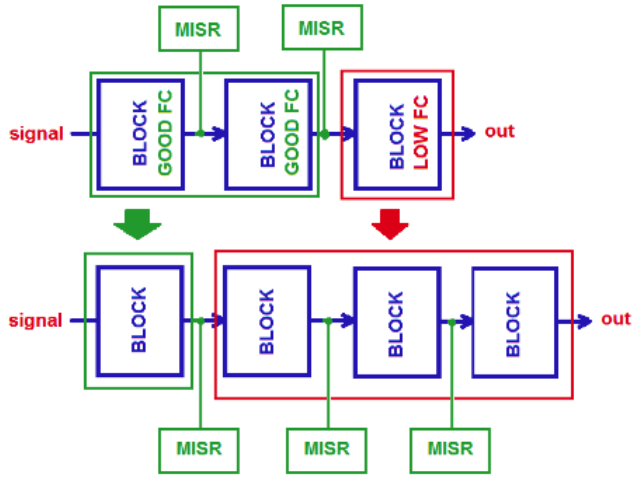



**Figure 3.4** – Example of merging and splitting the blocks in UUT with high and low fault detection coverage.

In the next Section a novel environment is presented that supports very high speed in analyzing the fault detection coverage in the blocks of UUT.

## 3.3 FAULT SIMULATION ENVIRONMENT

To carry out the procedure of minimizing the number of test-points according to Algorithm in Fig.3.3, large number of fault simulation sessions is needed for evaluating the fault detection coverage in the blocks of different size and for different partitioning solutions for the given UUT. A simple scheme for fault simulation of a sequential circuit is depicted in Fig.3.5. The model of the circuit and the test sequence form the input data for the simulator that calculates the fault detection rate. The faults are simulated in this case one by one. Such a single fault simulation is very slow. On the other hand, faster methods for fault simulation, such as deductive or critical path tracing based fault analysis, cannot be used for sequential circuits; they are only applicable for combinational circuits.

To overcome the difficulties of fault simulation in sequential circuits a special approach is proposed to escape from the dependency on feedback loops. Assume, the full sequential circuit (or a sequential block as a part of it) can be presented as a set of combinational sub-circuits each of them having a MISR in the output. By logic simulation of the test sequence for all sub-circuits, the input sequences are calculated (are fixed during the logic simulation). All the combinational sub-circuits can be fault simulated now independently, because each of them has MISR that detects the faults in the related sub-circuit. If the circuit cannot be partitioned in such a way, a traditional slow fault simulator for sequential circuits have to be used.

In order to calculate and save test sequences for all of the registers of the sequential circuit I have developed a toolset that allows to use standard state-of-

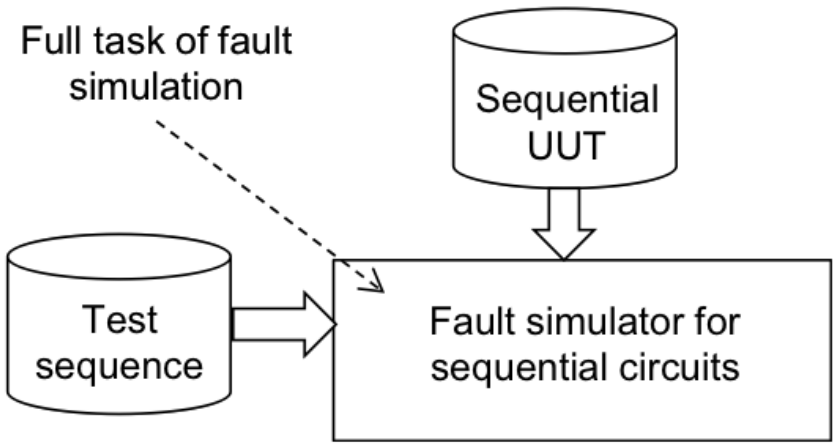



**Figure 3.5** – Fault simulation in sequential circuits.

the-art logic simulators for this purpose. The saving of register values is required in order to create a test set for combinational version of the design as was previously mentioned. The task is to allow register data to be saved into "XML" file for every clock cycle during logic simulation. Two separate tools were developed for this purpose - agmReader and tstCreator. The former is used before the logic simulation, and the latter - after the simulation.

The overall process of test set generation using sequential logic simulation is shown in Fig.3.6. The task of agmReader is to analyze the design and extract all necessary signals that have to be saved during logic simulation. It creates an additional "XML" descriptor file that contains the description of every registered signal. The description includes:

- Id
- Signal name
- Number of start bit
- Number of end bit

The signals are arranged in a way they appear in AGM model of the design. The AGM model is the combinational description of the circuit using Structurally Sythesized Binary Decision Diagrams (SSBDDs). Such a description has all the registers and feedback loops cut into separate inputs and outputs of the circuit. It has all the inputs of the registers as primary outputs of the circuit. Also all the outputs of the registers become primary inputs of the circuit. The ordering is important in the next steps of the process. The bit width of the signal is also important, as well as whether bit numbering starts from the LSB or MSB. Once the signals are analyzed the agmReader creates a description of VHDL process called

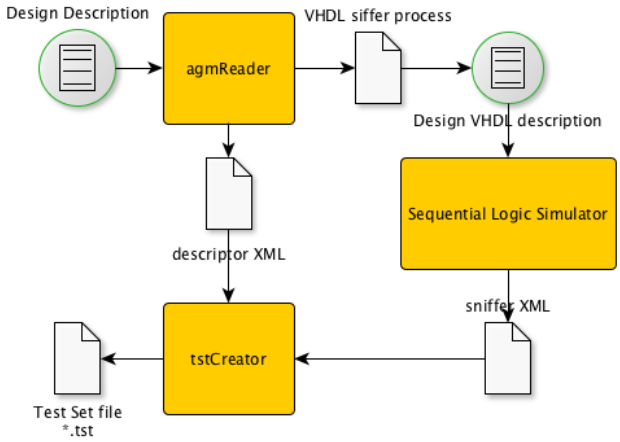



**Figure 3.6** – Extracting combinational test set using sequential logic simulation.

*sniffer*. This process contains a behavioral description allowing to save all the required signal values for every clock cycle. Once the tool has finished - VHDL sniffer process is integrated into VHDL description of the design.

As can be seen from the Fig. 3.6 the next step is to do a sequential logic simulation of the design. Any traditional logic simulator can be used for this purpose. During logic simulation the value of every signal for every clock cycle is saved into specially formatted "sniffer XML" file. Because the resulted file is a text file - it is important to keep the amount of auxiliary data as small as possible. This is why any information about the signals is not saved in it, except their values. In order to be able to decode the data after a simulation the signal values are saved in the same order they were analyzed and recorded to the "descriptor XML".

In order to remove the confusion that may arise at this point I recall. There are two "XML" files. The first file is a "descriptor XML", which is created during analysis of the design and contains information regarding the registered signals of the design and their specific ordering. Another file - "sniffer XML" is created during sequential logic simulation of the design and contains the values of all registered signals for every clock cycle. The values are written in the same order they appear in the "descriptor XML".

Once the logic simulation is finished the "sniffer XML" is saved. tstCreator tool is used to extract the data from it and create a test set file for combinational fault simulation. This tool reads both "descriptor XML" and "sniffer XML". Because "sniffer XML" contains data in integer format, using signal size and alignment information from descriptor the tool decodes the integer values into binary and aligns accordingly. As a result a saved data becomes a test set for combinational representation of the design.

These two tools are written in Java and form a part of a novel simulation environment. In this environment the fault simulation has to be carried out only in the combinational parts of the UUT. It is depicted in Fig.3.7.

The fault simulation is carried out in the following flow:

1. In each current step of the Algorithm in Fig.3.3, the UUT is partitioned into a set of blocks $S = SC \cup SS$ where $SC$ is a subset of combinational blocks and $SS$ is a subset of sequential blocks.

2. The UUT is simulated for the whole test sequence $T$, and for each block $B_i \in S$, the whole local subsequence $T_i$ at the input of $B_i$, caused by $T$ will be collected and stored. The subsequence $T_i$ will be regarded thereafter as the sub-test sequence for the block $B_i$ generated on-line by the test sequence $T$.

3. All the combinational blocks $B_i \in SC$, will be fault simulated for the local sub-test sequences $T_i$ with the fast fault simulator for combinational circuits as shown in Fig.3.7.

4. All the sequential blocks $B_j \in SS$, have to be fault simulated for the local sub-test sequences $T_j$ with the slow fault simulator for sequential circuits in this environment according to scheme Fig.3.5.

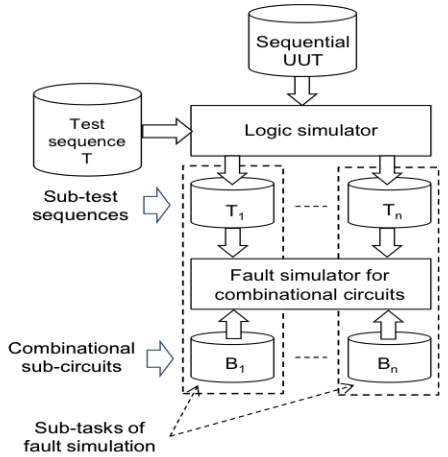



**Figure 3.7** – Transforming sequential fault simulation into sub-tasks of combinational fault simulation.

For fault simulation of combinational circuits a very fast fault simulator that implements a method of exact parallel critical path tracing, was used. It was chosen because of its good performance that is higher than currently used commercially available fault simulators[70].

The high speed in this simulator is achieved by reasoning the faults along signal paths in the circuit for $N$ test vectors in parallel, where $N$ is the number of bits in the computer word. The simulator runs in two sessions through the whole circuit. The first session is carried out only once for all the test vectors to be simulated. The goal of this session is to create a compact computing model for further fault reasoning that consists of a sequence of Boolean formulas. Since the formulas are Boolean, they can be processed in parallel. The second session is to calculate the detected faults for packages of N test vectors in parallel using the computing model created in the first session.

The simulator was included into the fault simulation environment in Fig.3.7, where it will be used for simulating faults in the blocks $B_i \in SC$, block by block.

Unfortunately, the simulator cannot be used for calculating the fault coverage for the sequential blocks $B_j \in SS$.

In the next Section, a case study will be discussed where I investigate the feasibility of the proposed method of at-speed self-testing in a pipe-lined signal pre-processor, described in previous chapter. Here I am going to compare the difference in performance of two fault simulation schemes depicted in Fig.3.5 and Fig.3.7. The results of fault simulation for the whole family of 8 processors (column 1) are presented in Table 3.1.

Column 2 describes the time in seconds for logic simulation of the sequence of 10 000 vectors on these benchmarks given by their behavior VHDL descriptions. The columns 3 - 6 describe fault simulation experiments according to Fig.3.7 on

the same sequence of 10000 vectors. In case of combinational fault simulation two levels of fault simulation are compared – gate-level and macro-level, where each macro represents a fan-out-free region (a gate-level sub-circuit) in a simulated combinational block of the given UUT. Only Stuck-at-Faults (SAF) were simulated.

However, to save the time, only the correct behavior was considered during behavioral level simulation (column 2). The real sequential fault simulation using fault injection can be much slower, as it requires to be run on gate-level representation that takes more time to execute. So that the results presented on the basis of behavioral logic simulation should be treated as ideally fast if used to represent sequential gate-level fault simulation. It is also worth mentioning that time values presented in column 2 in Table 3.1 are for single fault simulation. In order to obtain the fault simulation time for the whole circuit this value have to be multiplied by number of faults in the circuit (column 3 and 6). Time values in columns 4 and 7 are already presented for all the faults.

To compare the two fault simulation approaches presented in Fig.3.5 and Fig.3.7 on the basis of Table 3.1, lets consider the results for the processor architecture 8a (the 1st row). For simulating 112034 gate-level SAF faults using parallel critical path tracing in the environment in Fig.3.7 30 seconds is required. Assume now very optimistically that for single fault simulation of sequential circuits in

**Table 3.1** – Comparison of two fault simulation approaches.

| Circuit | Beh. level logic simulation, Single fault [sec] [Fig.3.5] | Fault simulation. All faults [Fig.3.7] | | | | | |
|---|---|---|---|---|---|---|---|
| | | Macro level | | | Gate Level | | |
| | | # of faults | Sim-n time, [sec] | Speedup | # of faults | Sim-n time, [sec] | Speedup |
| 8a | 0.155 | 66328 | 14.0 | 734 | 112034 | 30.0 | 579 |
| 8b | 0.152 | 50206 | 12.1 | 631 | 83940 | 24.7 | 517 |
| 8be | 0.168 | 55270 | 28.3 | 328 | 99330 | 62.1 | 269 |
| 8bk | 0.159 | 49938 | 11.6 | 684 | 86878 | 25.2 | 548 |
| 8bs | 0.154 | 56444 | 65.2 | 133 | 100820 | 173.4 | 90 |
| 8c | 0.159 | 73182 | 16.3 | 714 | 122386 | 35.9 | 542 |
| 8d | 0.161 | 71730 | 17.0 | 679 | 123012 | 35.5 | 558 |
| 8de | 0.164 | 75840 | 34.8 | 357 | 136876 | 81.3 | 276 |

the UUT at the behavioral level the same time is needed as for simulation of the correct circuit, i.e. 0.155 s. Then, to simulate 112034 faults in the sequentially presented gate-level UUT 17365 seconds, or about 5 hours would be needed. Hence, the gain in speed for this particular UUT will be not less than 580 times. In fact, it will be even more, since the gate-level simulation would be much slower than the behavior level simulation, as was already mentioned.

The results of benchmark 8bs deserve a separate discussion. As can be seen this benchmark has considerably smaller speed-up factor, than all the other circuits. It can be seen that in case of 8bs time of logic simulation is merely equal to the results of other circuits, but the time of combinational fault simulation is considerably higher than others. The reason lies in the internals of this circuit. 8bs is a modification of the bio-impedance analyzer that has maximum shared resources and therefore great number of reconvergent fan-outs. As it was previously shown [39] the exact critical path back-tracing used in fault simulation environment is very sensitive to the amount of reconvergency in the circuit. I can conclude here that logic simulation doesn't depend on the level of reconvergency in the circuit, apart from combinational fault simulation using back-tracing.

In order to produce the results in Table 3.1 I used desktop class Intel I7-930 @ 2.80 GHz 4-core processor running Windows 7 operating system with 6GB of physical RAM. The circuit was simulated by ModelSim SE ver.6.5c. The speedup values were calculated in respect to theoretically assumed speed of sequential fault simulation computed as multiple of column two and respective column for macro- and gate-level number of faults.

Note, the main idea of such a powerful fault simulation, based on transforming sequential fault simulation task into a set of combinational fault simulation sub-tasks is directly related to the goal of this analysis. And the idea is as well closely tailored in the method of at-speed testing being evaluated. The goal of fault simulation is in this case to evaluate the fault detection coverage, not fault diagnosis. In other word, I am not interested in creation of an exact fault table. As there are signature analyzers on the outputs of simulated blocks, it will be sufficient during testing to fix on the inputs of the block correctly only the first erroneous vector affected by the fault. As the result, the method is not sensitive to the possible mismatches of the subsequent input vectors of the faulty block with those collected during logic simulation of the correct UUT.

The main advantage of using such fault simulation as a part of the method proposed in Section 3.2 is to provide a designer with a good insight into the testability of the given circuit. Having an information regarding fault coverage of every combinational block the designer can improve testability more efficiently. He can limit the circuit space for testability improvement only to those blocks that have lower fault coverage. And for the blocks with good coverage it will be possible to merge them in order to cut the number of points for monitoring. Although for this task the sequential simulation is required - the size of the circuit to be simulated is significantly reduced. Also in the Chapter 3 I propose a general method to extend the usability of the PPECPT for simulation of the sequential

circuits, even in case of partial register monitoring. This new method can also be effectively used for the purpose of simulation of combined blocks.

Let us summarize the main idea of the section. The fast fault simulator is used [39] that is not simulating faults one by one like in the traditional fault simulators for sequential circuits, rather it calculates by a single run all the faults detected in the combinational sub-circuits by a bunch of patterns (the reasoning is done for all faults in the sub-circuits in parallel for many patterns). The confusion may arise now because the fault reasoning is carried out for input patterns that were collected from the behavior of the correct circuit. This means that if there was a fault, which produces an erroneous output pattern, then the next input pattern will be as well erroneous (because of the possible feedback loop) that means in turn that the results of fault reasoning of all subsequent patterns will be as well wrong. But, on the other hand, this is not any more important, because the first erroneous pattern in the input sequence of the sub-circuit will be fixed already by MISR as an error, and this will be sufficient for fault detection in the end of the test (with the accuracy determined by the probability of signature aliasing). Generating fault tables and fault diagnosis of course is not possible, but this is not the purpose of this chapter.

To my knowledge, such an approach of running fault simulation in sequential circuits for at-speed test has been proposed the first time.

## 3.4 CASE STUDY: SIGNAL PROCESSING UNIT AS UUT

To investigate the feasibility of the method in the sense of achieving sufficient fault coverage in real cases, I carried out experimental test research on one of the benchmark circuits previously proposed. From the family of processors, discussed in the previous Section the processor with architecture 8a was selected. It has been chosen because is represents design solution used in real-life scenarios.

As it was already mentioned in Chapter 3 the bio-impedance signal analyzer represented by circuit 8a is only a part of industrial solutions for bio-impedance measurements. The system also has a sine-wave generator of different frequencies. It is used to produce the excitation of a sine signal into a tissue in order to analyze it's response for further calculation of bio-impedance. During this case study in addition to evaluation of at-speed methodology it was also decided to check whether digitized analog sequence can be used as a test set. The reason for that is simple. In case such a sequence shows good fault detection quality a generator already present in system can also be used for test purposes. This would in turn result in savings of silicon area.

For self-test purposes, the analog part - DAC, tissue and ADC - are skipped and the output of the Excitation Signal Generator is fed directly to the digital input of the Digital Signal Analyzer that is shown in Fig.3.8.

Excitation signal generator along with body and analog part was exchanged with signal generator of particular type. Also the sampler is implemented as 80MHz clock signal. As was mentioned in Chapter 3 80Mhz is a maximum clock frequency of this analyzer. Fig.3.8 gives visual representation of the test setup.

I investigated four types of signal generators to be used as test sources: sine, chirp, saw-tooth, LFSR. Such signal generator provides input signals to all 8 channels of the analyzer. These can be seen on Fig.3.1. All the channels get the same signal, so that each channel can be tested equally to each other. All the generators are implemented in VHDL in order to be used with simulation environment.

1. The sine signal generator is using floating point arithmetic and sin() function of the VHDL math library. It can take amplitude, phase and frequency as parameters to produce the corresponding sine wave. During the experiments the amplitude was set to 15 bits, taking into account 1 sign bit and 16-bit wide input of the analyzer. The phase was set to 90 degrees in order to produce the input signal from the upper part of the wave. Such signal would produce more unique values in less time, because it covers all the values from top to the bottom in half-period. It was useful to check whether the test sequences of small length could produce meaningful results. The frequency was modified during the experiment in order to detect the better signal for testing this device.
2. The chirp generator takes as parameters start and stop frequency periods as well as number of samples in which frequency should change from start to end frequency value. The chirp generator changes the frequency every sample it produces. The amplitude remained 15bits + 1 sign bit and start phase was set to 90 degrees. During the experiments the length of the chirp signal was manipulated – number of samples from start to end frequency.
3. Saw-tooth signal is implemented as a counter. The parameter it takes is a period of the signal. The generator produces equally spaced samples of the saw-tooth signal of this period. The amplitude is 15bits+1 sign bit.

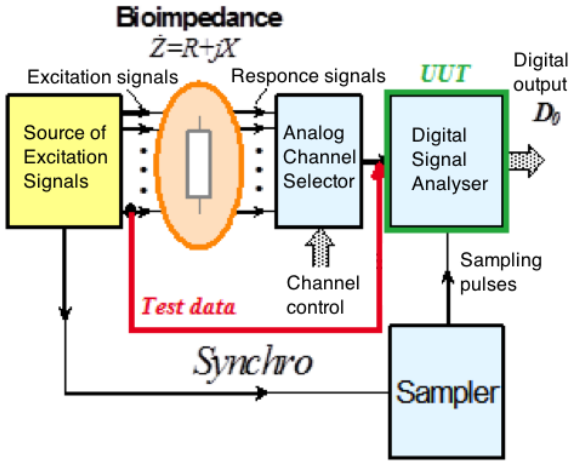



**Figure 3.8** – Testbench for the case study.

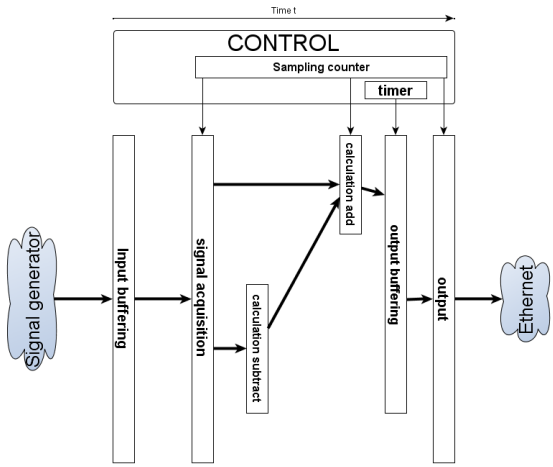



**Figure 3.9** – Equivalent circuit for the Signal Analyser in Fig.2.2.

4. LFSR signal generator is implemented as 16-bit linear feedback shift register. The seed is taken so that it goes through all the 65535 possible values except 0. The size of the LFSR was chosen in accordance to the input width of the signal analyzer under test.

## 3.5 EXPERIMENTAL RESULTS

Experiments were carried out for Signal Analyzer (architecture 8a) in Fig.3.8, presented as equivalent circuit with highlighted pipe-lined tracks in Fig.3.9. As the result of the experimental research according to method in Fig.3.3, the circuit was finally partitioned into 7 blocks as separate UUTs that are characterized in Table 3.2.

### 3.5.1 *Fault coverage of the blocks*

I calculated the fault coverage for all seven blocks as well as total fault coverage for four different types of signals: sine, chirp, saw-tooth and LFSR. The number of test patterns used for this simulation is 1000000 (one million). The fault simulation environment described in Section 3.3 was used. The results of the experimental research in percentage of fault coverage for all the different blocks are presented in Table 3.3 and as the bar diagram in Fig. 3.10.

Blocks timer and sampling represent control logic of the circuit. These are well tested, because their memory cells are completely covered by MISRs. The reason, why the coverage is not 100% is that the reset logic wasn't simulated.

As can be seen, the best results in average for all the blocks were achieved for the input signal sine where the fault coverage was 98.20%. The lowest total fault coverage 75.99% was registered for the signal type saw-tooth.

**Table 3.2** – Characteristics of the blocks in Fig.3.8

| No | Name of the block | Number of faults | Number of inputs | Number of outputs |
|---|---|---|---|---|
| 1 | calc_add | 69544 | 1431 | 896 |
| 2 | calc_sub | 18588 | 791 | 256 |
| 3 | in_buf | 98 | 17 | 16 |
| 4 | out_buf | 14750 | 1554 | 769 |
| 5 | out | 7480 | 709 | 64 |
| 6 | sig_acq | 8560 | 538 | 520 |
| 7 | timer | 512 | 18 | 17 |
| Total | | 119532 | 2528(5058) | 2538 |

**Table 3.3** – Results of fault coverage experiments.

| No | Name of the block | Input signal types | | | |
|---|---|---|---|---|---|
| | | Sine, % | chirp, % | saw-tooth, % | LFSR, % |
| 1 | calc_add | 97.37 | 94.86 | 76.80 | 95.71 |
| 2 | calc_sub | 98.85 | 99.20 | 64.90 | 99.20 |
| 3 | in_buf | 82.65 | 82.65 | 82.65 | 82.65 |
| 4 | out_buf | 99.88 | 99.86 | 74.74 | 99.86 |
| 5 | out | 99.14 | 99.06 | 78.66 | 99.14 |
| 6 | sig_acq | 95.63 | 95.63 | 95.63 | 95.63 |
| 7 | timer | 94.14 | 94.14 | 94.14 | 94.14 |
| 8 | sampling | 95.62 | 95.62 | 95.62 | 95.62 |
| Total | | 98.20 | 96.68 | 75.99 | 97.21 |

Considering the distribution of fault coverage among different blocks it can be seen that the lowest test quality is mapped to the block in_buf. However, since the block in_buf is rather small (characterized by only 98 faults), the improvement of its testability will not lead to considerable increase in the total fault coverage of the whole circuit.

### 3.5.2 *Impact of the test length*

Since the cost of testing depends on the time used for carrying out the self-test procedure, I investigated how the fault coverage will depend on the test length measured in the number of test patterns. The results are shown as the charts for four signal types in Fig. 3.11.

The most cost effective would be the LFSR based self-test sequence where the fault coverage around 90% will be achieved already after 80 000 test patterns (clock cycles) whereas the sine signal based and chirp signal based tests achieve only about 85% and 80% fault coverage, respectively, at the same test length. When doubling, however, the test length, the sine based and LFSR based tests become equal at the 95% fault coverage. Especially sensitive to the length of the test is the chirp signal based test sequence.

### 3.5.3 *Comparison to the state-of-the-art methods*

I compared the test quality achieved by the proposed method with traditional scan-path (SP) techniques both for using LFSR pseudorandom and deterministic test sequences. The results are presented in Table 3.4.

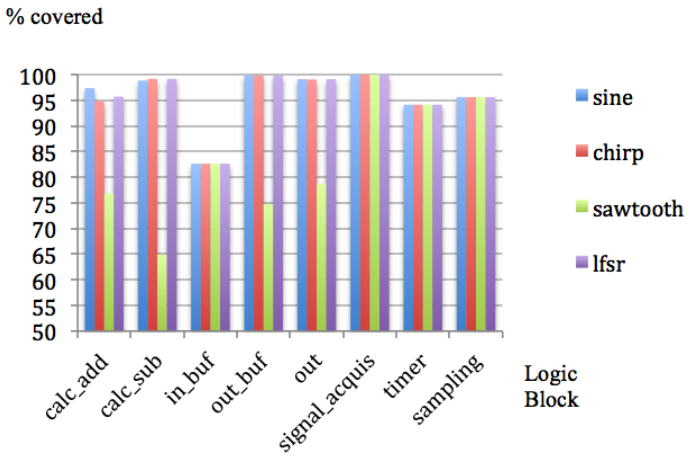



**Figure 3.10** – Distribution of fault coverage in the circuit.

Table 3.4 shows that the fault coverage is nearly the same for all the methods compared. However, to get the same fault coverage as with the proposed method, the test length of the scan path & LFSR based approach should be even twice bigger compared to the proposed method. To calculate the testing time cost in clock cycles, the test length for both referenced scan-path based methods should be multiplied by the length of the scan path that is equal to 2528 bits (the total number of inputs of all the tested blocks in the given circuit).

For the proposed method, the testing time in number of clocks is equal to the test length. Sp that, I can conclude that the time cost of the proposed method is about 3-7 times cheaper than the SP & deterministic approach and more than 2500 times cheaper than SP & LFSR at the same fault coverage (in the latter case the single scan-path was assumed).

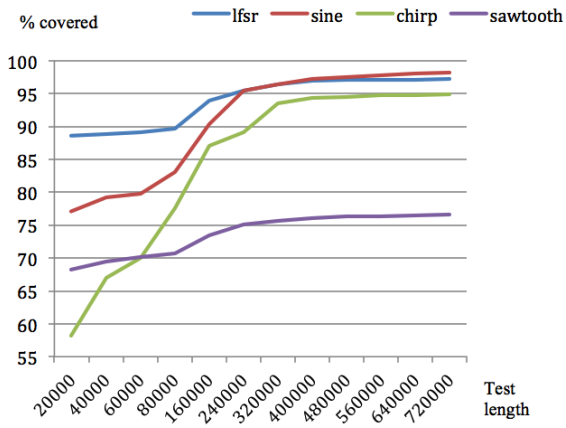



**Figure 3.11** – Dependence of the fault coverage on test length.

**Table 3.4** – Comparison of different methods.

| Method | Fault cover. % | Test length (TL) | Testing time (clock cycles) |
|---|---|---|---|
| Proposed | 97.78 | 500000 | $5 * 10^5$ |
| SP & LFSR | 96.82 | 500000 | $12640 * 10^5$ |
| Proposed | 98.20 | 1000000 | $10 * 10^5$ |
| SP & LFSR | 98.73 | 1000000 | $25280 * 10^5$ |
| SP & deterministic | 98.69 | 1364 | $34 * 10^5$ |

**Table 3.5** – Fault coverage for circuit 8a - without resource sharing and 8bs - with resource sharing.

| Signal | Fault coverage, % | |
|---|---|---|
| | 8a | 8bs |
| Sine | 98,20 | 98,16 |
| Chirp | 96,68 | 96,49 |
| Sawtooth | 75,99 | 81,69 |
| LFSR | 97,21 | 97,05 |

### 3.5.4 *Impact of the resource sharing*

In order to see how resource sharing would affect the proposed method I've computed fault coverage for another bio impedance benchmark circuit with the highest reconvergency characteristics that is 8bs. The number of test vectors is 1 000 000. The results of the fault simulation of circuit 8a and 8bs for all four analog signals can be seen in Table 3.5.

Although the results of the most test signals are lower for circuit 8bs, it can be seen that method still provides high fault coverage even in case of high level of reconvergency. The digitized sine signal still provides better results, than any other test signal. The results of saw-tooth test signal improved that suggests that circuits with higher resource sharing are better testable with exhaustive test sequences. However its results are still low, when compared to other digitized analog signals. These preliminary results show that the efficiency of proposed at-speed functional BIST methodology doesn't considerably depend on the amount of resource sharing.

## 3.6 CHAPTER SUMMARY

In this Chapter the following main results were achieved:

- A new approach was developed to self-testing of digital systems with pipelined architectures using inherent functionalities of systems with added value as higher test quality, less hardware cost, and removing yield loss due to avoiding the danger of overtesting.
- The usage of digital representation of analog signal sequences as a functional test for testing digital circuits (signal processing architectures) is investigated in the first time.
- A novel BIST evaluation environment was developed that allowed a gain of 580 times in speed of fault simulation.

The added value of using inherent functional self-test sequences is the higher test quality explained by on-line at-speed testing. The approach to functional

BIST does not need to store high volume test data in the system memory. Additional hardware is as well not needed for on-line test pattern generation as in the case of traditional LBIST. The only needed additional test hardware is related to using MISR for monitoring the test responses. To minimize the needed additional MISR hardware overhead, an original algorithm for selecting test-points was developed. As the result of avoiding artificial embedded test pattern generators like in case of LBIST, and of using only normal working sequences for test purposes, the dangers of over-testing and the related yield loss are removed.

To cope with the problem of very slow fault simulation in sequential circuits needed for exploration and comparison of different self-test solutions a novel evaluation environment was developed where the time consuming sequential fault simulation task can be transferred into a set of combinational fault simulation sub-tasks. Experiments demonstrated the gain in evaluation speed more than 580 times without losing any accuracy in fault coverage calculation.

To investigate the feasibility of the method to achieve high fault coverage, experimental research with a digital Signal Analyzer unit was carried out as a case study.

The goals of the experiments were twofold: (1) to select the best type of input signal for testing purposes from a set of signals typically used for processing in the given Signal Analyzer, and (2) to compare the new method with traditional scan path based testing methods.

Experimental research showed that the best testing capability has the sine signal (with fault coverage of 98.2%) compared to the LFSR based pseudorandom (97.2%) and chirp (96.7%) signals at the same test length. The worse testing capability has the saw-tooth type signal (76%). The fault coverage achieved by the sine signal was 98.2% that is nearly the same compared to the traditional scan-path pseudorandom (98.7%) and deterministic (98.7%) test approaches. The gain in testing time cost was 3-7 times compared to the deterministic and more than 2500 times compared to the pseudorandom single scan-path based approach.

The comparison of the results for circuits with different level of resource sharing suggests that methodology is not sensitive to this parameter, at least for the given benchmark.

To be efficient the methodology requires strong evaluation environment based on fault simulation. The PPECPT algorithm used in this work has very limited use in case of sequential circuits. It also has a potential for further speed up in fault simulation of combinational circuits. In the next two chapters I propose methods to speed up the PPECPT algorithm based fault simulation for both combinational and sequential circuits.

# 4

# MULTI-CORE FAULT SIMULATION ENVIRONMENT

This chapter is based on the publication "Fault Simulation with Parallel Exact Critical Path Tracing in Multiple Core Environment" (see Appendix D).

In this chapter a novel fault simulation method is proposed, based on exact critical path tracing beyond the Fan-out-Free Regions (FFR) throughout the fully simulated circuit. The method exploits two types of parallelism: bit-level parallelism for multiple pattern reasoning, and distribution of the fault reasoning process between different cores in a multi-core processor environment. To increase the speed and accuracy of fault simulation, compared with previous methods, a mixed level fault reasoning approach is developed, where the fan-out re-convergence is handled on the higher FFR network level, and the fault simulation inside of FFRs relies on the gate-level information. To allow a uniform and seamless fault reasoning, Structurally Synthesized BDDs (SSBDD) are used for modeling on both levels. Experimental research demonstrated very promising results in increasing the speed and scalability of the method.

The contribution of the author lies in developing of the proposed algorithms, implmentation of required tools and carrying out the experiments.

The rest of the chapter is organized as following. Section 4.1 provides an overview and comparison of the available single-core and multicore fault simulation methods. Section 4.2 explains the theory behind the Parallel Pattern Exact Critical Path Tracing (PPECPT) algorithm. In Section 4.3 a new method of parallel critical path tracing based on mixed level fault simulation with two types of SSBDDs is proposed. Section 4.4 describes the method of distributing the task between multiple cores of the processor. Finally Section 4.5 describes the results of experimental research with related discussion, and Section 4.6 concludes the chapter.

## 4.1 OVERVIEW

Fault simulation is one of the common and challenging tasks in nowadays test process. In case of combinational circuits if the faults are simulated one by one like in case of serial fault simulation the number of operations depends on number of faults, number of test patterns and a size of the circuit. Number of faults is roughly the same as the number of gates in the circuit. Therefore if I denote number of gates as $n$ and number of patterns as $p$ - the complexity of the fault simulation roughly becomes $O(pn^2)$, which can be challenging for large

circuits. This is why a lot of work have been dedicated to improve the complexity and speed of the fault simulation process.

### 4.1.1 *Serial Fault Simulation*

It is a simplest way of fault simulation of the circuit. It consists of fault-free simulation for all test vectors available, to obtain "good" output responses and many simulations of "faulty circuits" to get "bad" responses. The faulty circuit is the one where a single stuck-at fault is injected at a time. If the output responses of faulty circuit do not match to the corresponding responses of fault-free circuit the fault is said to be detected by a given test vector.

Prior to fault simulation the list of faults is collapsed. The collapsing is possible due to the fact that some faults at the ouput of the gates depend on some faults on the inputs of these gates. It means that it is only sufficient to run fault simulation for one of these two faults. Fault collapsing is said to reduce the number of faults by 50% to 60% [13]

If it is only required to find the detected faults during fault simulation then it is sufficient to partially simulate the faulty circuit most of the time. It means the fault simulation is performed for faulty circuit with particular fault injected up to the moment, when it becomes detected. This approach is called "fault dropping". As many faults are detected after relatively small number of test vectors applied it can dramatically improve the performance of the method. Although quite efficient the use of this technique can be limited. Fault dropping should be avoided when fault simulation is used for diagnostic purposes. Also N-fault simulation, where the particular fault requires to be detected no less than N times, can significantly degrade the boost in performance.

The main disadvantage of the Serial Fault Simulation is its low performance. The advantages are simplicity and universality. Simplicity means it is only required to have standard logic simulator with fault injection and response comparison procedures. Also the ability to support different fault models, as long as the fault can be injected provides universality.

### 4.1.2 *Parallel Fault simulation*

Parallel fault simulation can be divided in two groups: parallel-fault and parallel-pattern. These two types of fault simulation take advantage of the CPU word-length, which is commonly 32- or 64-bit wide. The 64-bit processor can process logic operations for 64 bits at once. This parallelism can be realized in two ways: to simulate faults in parallel (parallel fault simulation) or patterns (parallel-pattern fault simulation).

In *parallel-fault simulation* the word length $w$ is divided between $w-1$ faulty circuits. One of the bits is dedicated for fault-free simulation. Another $w-1$ bits represent signal values at presence of different faults. When one of the faulty gates is reached the bit corresponding to this particular fault is set to the faulty

value. As digital circuit consists of logic gates the corresponding bitwise logic operation is used to simulate signals in parallel. When the faulty response is obtained at the primary output of the circuit the bits not equal to the fault-free bit show that particular faults were detected by current pattern at this particular output.

The performance boost in comparison to serial fault simulation is about $w - 1$ times. However it comes at a cost of lack of universality and limited fault dropping capabilities. In terms of universality the zero or unit delay models can only be used, as several faults are computed in parallel. By the same reason we can only drop faults, when all the faults from the bunch are detected.

In *parallel-pattern fault simulation* the word length $w$ is used to pack $w$ test vectors to simulate one faulty circuit for all of them in parallel. The approach also carries another name of "Parallel-Pattern Single Fault Propagation" (PPSFP).

The fault simulation process is as following. The test patterns are combined into a bunches of CPU word-length. The fault is injected and all of the bunches are simulated against this faulty circuit. If the fault dropping is used the simulation ends once the fault is detected or all of the bunches are simulated. In order to reason about fault detection the primary output responses are compared against fault-free simulation responses. If some of the patterns in the bunch produce different response it means they detect this particular fault. Once the fault is processed for all available test patterns the simulation repeats for other fault, until all the faults are simulated.

PPSFP approach better handles fault dropping in comparison to parallel fault simulation, as it can drop fault right away once it has been detected [62].

### 4.1.3 *Deductive fault simulation*

This type of fault simulation differs from the previously described techniques in a way that it uses a logic reasoning, rather than simulation. It is capable to produce a list of faults detectable by a given test vector in one run.

The reasoning starts at the primary inputs and "propagates" a list of faults through the circuit to the primary outputs. During this process the list of faults, which is capable to influence the fault-free value of a signal is computed for every signal including primary outputs. As a result at the end of the simulation process the faults located in the lists of primary outputs are said to be detected by a given pattern.

Fig. 4.1 shows the process of deductive fault simulation for a simple circuit for one test vector. A letter designates every line in the circuit, including branches of the fanout. Starting at the inputs of the circuit it can be seen that only stuck-at one fault (A/1) can change the value of 0 at line A. The same situation is at primary input B. However line C can be influenced not only by fault C/0, but also by preceding B/0. Looking at AND gate G1 it can be seen that both B/0 and C/0 could not propagate further, because value of 0 at line A blocks the propagation. On the other hand value of 1 at line C allows fault A/0 to propagate to the output

of G1 to line E. So that fault list of E contains A/0, along with E/1, but not B/0 or C/0. Line D can also be influenced by fault B/0, as well as D/0. Looking at OR gate G2 reveals that both A/1 and E/1 are blocked by D value of 1. While at the same time E value of 0 allows B/0 and D/0 propagation to the output of the gate to the line F. The fault list of line F would eventually consist of fault F/0 as well as B/0 and D/0, propagated through G2. These three faults can be marked as detected by test pattern "0,1". The similar reasoning is done for all of the test vectors.

The method is efficient because it only requires trivial logic simulation to be run. On the other hand this method also has some limitations. First of all there are undefined memory requirements, as the size of fault lists cannot be predicted in advance. Another limitation is the ability to handle only zero-delay timing model, as no timing information is considered during reasoning process [62].

### 4.1.4 *Concurrent fault simulation*

This type of fault simulation exploits the fact that only a fraction of the circuit in the output cone of the fault is influenced. So that it uses event-driven simulation to only additionally compute events, which differ from the fault-free simulation events. All the faults are concurrently processed along with fault free simulation for one pattern at a time.

The simulator processes good and bad events in the same run for one test vector at a time. The good events are those, which occur in case of fault-free behavior. The bad events occur in case when faulty behavior of a gate produces the result, which differs from fault-free behavior.

Every gate contains a fault list consisting of bad gates - gate copies with particular fault present. Bad gates are described by fault index and I/O values in the presence of a fault. Bad gates are only simulated when become visible. It happens when certain fault produces bad event, which propagates to this gate.

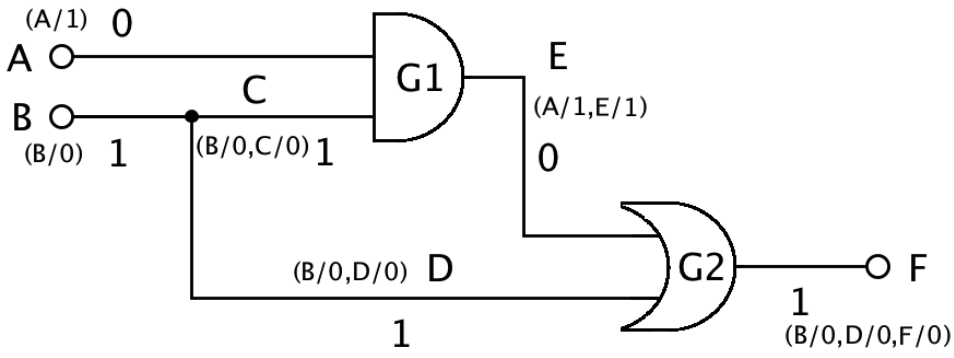



**Figure 4.1** – Deductive fault simulation.

The beginning of fault propagation is called divergence of a fault and the end of propagation - a convergence of a fault. When the fault diverges is create a bad event on the inputs of the next gates connected to the output of the faulty gate. When this happens the bad copies of these gates corresponding to this fault become visible and will be computed. When bad gate produces the same result as a good gate - it converges, so that it doesn't create a bad event for subsequent gates. In case the bad gate for this fault was present in the fault lists of those gates from previous run of the simulator - it is deleted. This way the only faulty behavior that is currently active for simulated test pattern is concurrently computed for all active faults.

The advantage of such approach is the performance improvement, especially for large circuits, where single fault could affect about 10% of the circuit. In this case the speed up would be an order of magnitude. On the other hand the handling of lists of bad gates requires a lot of dynamically allocated memory, which size is undefined in advance [62].

### 4.1.5 *Differential Fault Simulation*

This type of fault simulation is dedicated to improve sequential fault simulation by combining the good sides of serial fault simulation and concurrent fault simulation. Concurrent fault simulation has potential memory problem, due to simulation of all the faults at once, which is bad. However it uses previous state of the circuit - produced by previous pattern - to generate events for the next pattern, thus doesn't require storing and restoring the fault free state of the circuit, which is good. On the other hand the serial fault simulation simulates the faults one at a time and doesn't have undefined memory requirements, which is good. However it should keep the good state of the circuit in order to restore it every time the new fault is injected, which is bad. Differential fault simulation taked advantage of event-driven simulation of one fault at a time using circuit state from previously injected fault. This way it considerably saves memory.

First the fault-free circuit is simulated for the first test vector and the output response of the fault-free circuit is saved and the state is stored. The important fact is that only flip-flop values are stored, which reduces the memory requirements. Then the fault is injected and produces an event that is propagated for the same test vector to the primary outputs. The difference in state between previous state and current state of the flip-flops is stored for current fault. All the next faults are simulated in the similar manner for the first test vector and the corresponding consecutive differences of states are stored.

Next the new test vector is taken and fault-free state is restored from the memory. The circuit is simulated and output values are saved. Then the first fault is injected and the state of the flip-flops is restored using state difference saved during simulation of this fault for previous test vector. This way all the test vectors are simulated one by one.

If the faulty circuit outputs are different from the fault-free circuit output values - the fault is detected and can be dropped if required. The fault dropping is not trivial however, as it requires the state difference of dropped fault to be accumulated in to state differences of its consecutive undetected faults.

The shortcoming of the method is that the timing of the occurrence of events is not the same as their order. It means that in case when gate delays must be counted the memory requirements could become high [62].

### 4.1.6 *Critical path tracing*

The critical path tracing method [2, 6] eliminates explicit fault simulation for faults within Fan-out-Free Regions (FFR). Similarly to deductive fault simulation the reasoning about fault propagation from inputs to the output of the gate, macro or FFR is done, instead of true simulation of faulty circuit. However the reasoning process is accomplished in reverse direction: starting at primary outputs and finishing at primary inputs. The previously computed fault propagation information is constantly reused, which eliminates redundant computations.

The main challenge in such an approach is exact fault propagation computation for global reconvergent fan-outs. A modified critical path tracing technique that excludes fault simulation for fan-out stems and includes a system of rules to check the exactness of critical path tracing beyond the FFRs, and which is linear in time, is proposed in [79]. However, the rule-based strategy does not allow parallel analysis and rule check of many patterns simultaneously.

This drawback was removed in [68] by introducing a novel concept of Parallel Pattern Exact Critical Path Tracing (PPECPT) which can be applied efficiently also beyond FFRs. In [70], the same method was extended from stuck-at faults (SAF) for a general class of X-faults. The main idea of the method was in compiling of a dedicated compact computing model through the circuit topology analysis, which allows exact critical path tracing for many patterns throughout the full circuit and not only inside FFRs. The method is described in more detail in Section 4.2.

### 4.1.7 *Multi-core methods*

Uniprocessor methods make use of CPU word length for parallel computations. In [25] authors went further and made use of GPU to extend bit-level parallelism to thread-level parallelism, where multiple threads are computing different bunches of patterns in parallel (PP) for one fault. The approach is targeted to produce fault table. Fault parallel (FP) methods described in [53, 54] target distributed MIMD systems, such as hypercube computers from Intel. These approaches divide fault set into parts, which are distributed among computing nodes to simulate them pattern- by-pattern in parallel. Circuit parallel (CP) method described in [13] uses a set of Sun 3/60 workstations for distributed simu-

lation. The circuit is divided by levels, where gates inside the level are distributed among workstations for event-driven simulation in parallel.

There were also attempts to combine the two of the methods. The combinations of FP and PP have been proposed in [26, 30, 37]. In [30] the vector processor is used to dynamically balance the simulation yield, by making use of FP approach on the early stages of fault simulation in order to quickly cover easy-to-detect faults. Then the algorithm dynamically scaled down the fault parallelism, using more pattern parallelism at the end of simulation in order to quickly cover hard-to-detect faults. The SPITFIRE-2 described in [14] has two stages. In the first stage the test set and fault set are divided into parts and easy to detect faults in fault subsets are simulated using test subsets. In the second stage undetected faults are also partitioned into parts and pattern parallel fault simulation is carried out on those parts to cover hard-to-detect faults quicker. In SPITFIRE-3 the second stage is implemented as pipeline improving fault detection rate. It is important to note that this approach targets sequential circuits, as well as PAUSIM [26]. The major difference in PAUSIM is that the CPU word length is used to represent faults, so that bunch of 32 faults can be simulated for each test vector in parallel on single 32-bit processor. The test and fault sets are then divided into subsets and simulated on distributed system of UNIX workstations.

The combination of FP and CP have been proposed in [55] targeting sequential circuits. It also consists of two phases, where in the first phase easy-to-detect faults are covered by FP simulation. Later the second phase uses event-driven CP simulation to cover hard-to-detect faults.

PP and CP combination was proposed in [24]. The method is targeted for execution on GPU. Paper states that circuit level parallelism is used to simulate gates in the same level in parallel. One sub-processor only simulates 2 patterns simultaneously, due to the LUT based model of the gates used during simulation. The paper states that N test patterns could be simulated in parallel, but the details of implementation are not discussed. The method also can not produce fault table.

In the recent paper [34] it is shown that applying algorithmic optimizations and using parallelization on pattern and structural levels yield good results on multicore systems. Authors advanced Pattern Parallel Single Fault Propagation (PPSFP) concept using parallel graph computations and fault dropping and were able to achieve about 16x increase in speed.

The advantage of using model parallelism in multicore environment is also highlighted in another work [42]. Here the parallelism is achieved in three dimensions: algorithm parallelism (AP), circuit parallelism (CP) and fault parallelism (FP). The method is executed on GPU device from NVIDIA and is optimized to use local memory of multiprocessors. Only fan-out region affected by injected fault is processed. Faults influencing the same fan-out region are processed in parallel. Gate-level circuit representation is used. Gates of the same level are also processed in parallel. The approach achieves over 35x performance increase in comparison to traditional FSIM simulator [41]. Because the proposed method is an advancement of FSIM it is also using fault dropping.

Taking into account the possible advantages of exploiting the circuit parallelism on many-core system I describe the method to achieve better performance of exact critical path back-tracing in multi-core environment. The fault simulation can be accomplished on gate, macro or FFR levels. Two types of parallelism are utilized during fault simulation: (1) bit-level parallelism for multiple pattern reasoning originally present in PPECPT, and (2) distributing the compiled computing model among a subset of different CPUs in a multi-core computing environment, so that each processor is responsible for parallel critical path tracing in a related particular sub-circuit area. No fault dropping is used, so the method can be used for fault table generation.

Another novelty introduced in this chapter is a mixed level fault reasoning approach, where the problems related to the fan-out re-convergence are handled on the higher FFR network level, and the increased speed and accuracy in fault reasoning is achieved by fault reasoning inside FFRs using additional gate-level simulation data. Such an approach allows to achieve a speed-up in fault simulation inside FFRs and improve the accuracy of fault reasoning compared with previous methods in [68, 70].

## 4.2 CRITICAL PATH FAULT TRACING

Here I describe the method of critical path fault tracing using SSBDDs in more detail. I start with definition of Structurally Synthesized Binary Decision Diagrams (SSBDD). Then the fault simulation beyond the fanout stems is presented. FInally the high speed of fault simulation using SSBDDs is explained.

### 4.2.1 *Structurally Synthesized Binary Decision Diagrams*

Structurally Synthesized Binary Decision Diagrams (SSBDD) were introduced in [65, 67] as an extension of the traditional model of Binary Decision Diagrams (BDD). A BDD is a mean to represent, analyze, test and implement a Boolean function. It is defined in [14] as a directed acyclic graph with two terminal nodes, which are the 0-terminal and 1-terminal nodes. Every input variable of the Boolean function is represented by non-terminal node and has two outgoing edges, called 0-edge and 1-edge.

The formal definition presented in [65] is the following. An SSBDD that represents a Boolean function y = f(x) = f(x1, x2, ..., xn) is a BDD in which (inverted or non inverted) Boolean variables xi, (i= 1,2, ..., n) label nonterminal nodes, and constants 0 or 1 label terminal nodes.

SSBDDs are synthesized directly from the structure of the logic gate-level circuit. Superposition of elementary BDDs of the gates is used for this purpose. SSBDDs are used to represent fan-out-free regions, which are interconnected together into complete SSBDD model of the circuit. Because every combinational circuit can be regarded as a network of fan-out-free regions the SSBDD model

forms its equivalent representation. A side effect of the SSBDD model is its linear complexity and fault collapsing.

There are two types of mappings between SSBDDs and logic circuit:

- signal paths are represented by nodes of SSBDD
- groups of SSBDD nodes represents certain sub-circuits of the whole circuit

The major advantage of SSBDDs is that they represent the structure of the circuit in compact manner, as well as its behavior. This fact makes it possible to use single SSBDD model in a wide variety of tasks such as test generation, fault and logic simulations, testability improvement and fault analysis [65].

### 4.2.2 *Parallel pattern critical path fault tracing*

Consider a combinational circuit as a network of FFRs, where each of them is represented as a Boolean function

$$y = F(x_1, x_2, \ldots, x_n) = F(X) \tag{4.1}$$

where $X = x_1, x_2, \ldots, x_n$ is the input vector of the FFR. Such a network of 5 FFRs is represented in Fig.4.2. Let $X_k$ denote the vector of input variables of the k-th FFR, $z_k$ denote the internal fan-out stem variables (outputs of FFRs) with $z_{kj}$ as fan-out branch variables for $z_k$ (inputs of FFRs) and y denote the output variables of the circuit.

The fault simulation can be processed as calculation of Boolean derivatives: if $\partial y/\partial x = 1$ then the fault is propagated from $x$ to $y$. This check can be performed in parallel for a set of test patterns. In order to extend the parallel critical path tracing beyond the fan-out free regions the concept of Boolean differentials is used [63].

Consider the full Boolean differential of the FFR $y = F(X)$ as

$$\begin{aligned} dy &= y \oplus F((x_1 \oplus dx_1), \ldots, (x_n \oplus dx_n)) \\ &= y \oplus F(X \oplus dX) \end{aligned} \tag{4.2}$$

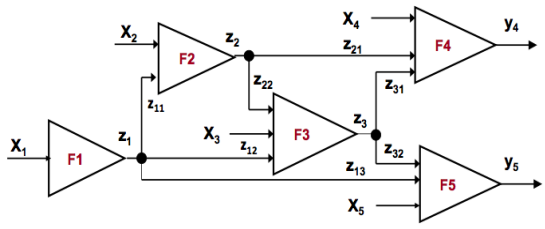



**Figure 4.2** – Combinational circuit with 5 FFRs.

The change of the value of $x$ because of the influence of a fault at $x$ is denoted here by $\partial x$. Also $\partial y = 1$ if some erroneous change of the values of arguments of the function (4.2) causes the change of the value of $y$, otherwise $\partial y = 0$.

In [68] it have been shown that from the expression (4.2) the following relationship can be derived:

$$\begin{aligned}\frac{\partial y}{\partial x} &= y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}dx), \ldots, (x_n \oplus \frac{\partial x_n}{\partial x}dx)) \\ &= y \oplus F(X \oplus \frac{\partial X}{\partial x}dx)\end{aligned} \tag{4.3}$$

The formula (4.3) taken in the vector form can be simplified as

$$\frac{\partial y}{\partial x} = y \oplus F(X' \oplus \frac{\partial X'}{\partial x}dx, X'') \tag{4.4}$$

where $X' \subset X$ is the sub-vector of variables which depend on $x$, and $X'' = X \backslash X'$ is the sub-vector of variables which do not depend on $x$.

For example, for calculating if the fault on $z_2$ can be detected on $y_4$, one can check if

$$\begin{aligned}\frac{\partial y_4}{\partial z_2} &= y \oplus F(X_4, z_{21} \oplus 1, z_{31} \oplus \frac{\partial z_3}{\partial z_2}dz_2) \\ &= y \oplus F(X_4, \overline{z_{21}}, z_{31} \oplus \frac{\partial z_3}{\partial z_2}dz_2) = 1\end{aligned} \tag{4.5}$$

The formula (4.4) can be used for calculating the influence of the fault at the common fan-out stem $x$ on the output $y$ of the converging fan-out region by consecutive calculating of Boolean derivatives over related FFR chains starting from $x$ up to $y$. For that purpose, for each converging fan-out stem, the corresponding formulas like (4.4) should be constructed for each converging FFRs. All these formulas will constitute partially ordered computation model for fault simulation. Since the formulas are Boolean, all computations can be carried out in parallel for a bunch of test patterns.

Introduce first the following notations for the formulas above which are used for calculating the Boolean derivatives:

- $(x, y)$ - for $\partial y / \partial x$
- $\{X_k, y\}$ - for a subset of formulas $\{\partial y / \partial x \mid x \in X_k\}$
- $R_{xy}((x, x_1), \ldots (x, x_k))$ - for the general case (3), where $X' = (x_1, \ldots, x_k)$
- $D_x$ - vector which shows if the fault at the node x is detected or not detected at any circuit output
- $DX$ - a set of vectors $D_x$ for the nodes $x \in X$

An example of a computational model of fault simulation for the circuit in Fig.4.2 is presented in Table 4.1.

The formulas presented in Table 4.1 can be easily created and stored by the topological tracing of the circuit by algorithms developed in [68]. The algorithm has linear complexity. However, the complexity of the computational model and the related fault simulation speed depends on the structure of the circuit. As shown in the papers [69, 70], the speed of the fault simulation by the proposed parallel critical path tracing method outperforms the speed of the fault simulators of major CAD vendors.

### 4.2.3 *Fast fault simulation with SSBDDs*

The high speed of processing the formulas is achieved by using SSBDDs for modeling FFRs [65, 66]. Each FFR $y = F(X)$ is represented by an SSBDD $G$, and

**Table 4.1** – Leveled fault model equations.

| L | Partially ordered formulas | Types of simulation tasks |
|---|---|---|
| 7 | $\forall x_{4,i} \in X_4 : D_{x_{4,i}} = \{x_{4,i}, y_4\}$, $Dz_{21} = (z_{21}, y_4), Dz_{31} = (z_{31}, y_4)$; $\forall x_{5,i} \in X_5 : D_{x_{5,i}} = \{x_{5,i}, y_5\}$, $Dz_{13} = (z_{13}, y_5), Dz_{32} = (z_{32}, y_5)$ | Fault simulation inside the FFRs (F4 and F5) |
| 6 | $Dz_3 = Dz_{31} \vee Dz_{32}$ | Fault simulation of fan-out stems (z3) |
| 5 | $\forall x_{3,i} \in X_3 : Dx_{3,i} = x_{3,i}, z_3 \wedge Dz_3$, $Dz_{22} = (z_{22}, z_3) \wedge Dz_3$, $Dz_{12} = (z_{12}, z_3) \wedge Dz_3$ | Fault simulation inside the FFRs (F3) |
| 4 | $Dz_2 = Rz_2, y_4((z_2, z_{21}) \equiv 1, (z_2, z_{31})) \vee ((z_{22}, z_{32}) \wedge Dz_{32})$ | Fault simulation of fan-out stems (z2) |
| 3 | $\forall x_{2,i} \in X_2 : Dx_{2,i} = x_2, i, z_2 \wedge Dz_2$, $Dz_{11} = z_{11}, z_2 \wedge Dz_2$ | Fault simulation inside the FFRs (F2) |
| 2 | $Dz_1 =$ $((z_1, z_3) \wedge Dz_{31}) \vee Rz_1, y_5((z_1, z_3), (z_1, z_{13}) \equiv 1)$ where $(z_1, z_3) = Rz_1, z_3((z_1, z_{22}), (z_1, z_{12}) \equiv 1)$ | Fault simulation of fan-out stems (z1) |
| 1 | $\forall x_{1,i} \in X_1 : Dx_{1,i} = x_1, i, z_1 \wedge Dz_1$ | Fault simulation inside the FFRs (F1) |

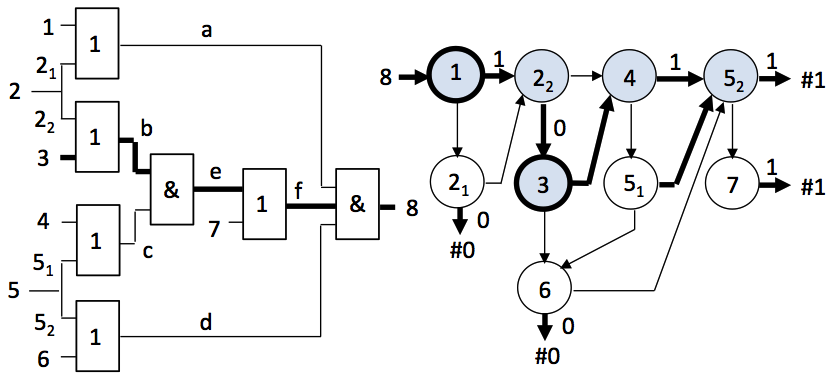



**Figure 4.3** – An FFR of a combinational circuit and its SSBDD.

each signal path in the FFR represented by a variable $x \in X$ is modeled by a corresponding node in the $G$. All the faults on a signal path collapsed into the faults on the inputs of the FFR, are modeled by the faults at the nodes in $G$. Hence, the targets of the fault simulation are the faults at the SSBDD nodes.

Consider a circuit in Fig. 4.3, and its corresponding SSBDD. The circuit contains nine signal paths, and a node in the graph represents each of them. Note, only the branches of the fan-out inputs are represented in the SSBDD as the model of the FFR. Fault simulation is carried out by traversing the nodes in the graph according to the given test patterns as in the case of traditional BDDs [47].

For simplification the graphical representation of SSBDDs, I use here the following convention: from a node labeled by a variable $x$, the right-hand edge corresponds to the value $x = 1$, and the down-hand edge corresponds to the value $x = 0$. Correspondingly, the exit from the graph to the right means entering the terminal node with constant #1, and the exit from the graph downwards means entering the terminal node with constant #0.

Consider a test pattern 1011101 (1234567) at the inputs of the FFR in Fig. 4.3. The pattern detects the fault at the input 3 by propagating the faulty signal from the input 3 to the output 8. On the SSBDD in Fig. 4.3 the edges activated by this pattern are highlighted in bold. The nodes traversed in the graph during simulation of the pattern are marked by gray color. The value on the output 8 of the circuit at this pattern is $y = 1$. Since the nodes 1, $2_2$, 3, 4, $5_2$ are traversed, all they are responsible for the value $y = 1$s, and hence, should be taken as fault candidates in case if the error will be noticed at the circuit output. All other nodes $2_1$, $5_1$, 6, and 7 have not contributed in fault simulation, and hence, can be excluded from the fault candidates set. Next, by simulating the faults at candidate nodes it can be easily noticed that only the faults at the nodes 1 and 3 are detected by the given pattern, because at these faults on the graph the terminal node #0 will be reached which means y = 0.

In [68], the algorithms for parallel logic simulation and parallel fault simulation on SSBDDs were proposed. The algorithms are based on the ordering of

nodes $m$ by assigning them numerical labels, so that for each node $m$ with label $n(m)$, all its predecessors $m_j$ must have labels $n(m_j)$ less than $n(m)$. Logic simulation is based on recursive calculating of the value of the formula

$$D(m) = (x(m) \wedge D(m^1)) \vee (\neg x(m) \wedge D(m^0)), \tag{4.6}$$

where $D(m)$ for the terminal nodes is equal to the respective constants #1 and #0. Here $x(m)$ denotes the node variable, $m_1$ and $m_0$ are the neighbors of $m$ in directions of $x(m) = 1$, and $x(m) = 0$, respectively. Fault simulation is based on recursive calculating of values of the formulas

$$L(m^1) = L(m^1) \vee (L(m) \wedge x(m)), \tag{4.7}$$

$$L(m^0) = L(m^1) \vee (L(m) \wedge \neg x(m)), \tag{4.8}$$

$$S(x(m)) = \frac{\partial y}{\partial x(m)} = L(m) \wedge (D(m^0) \oplus D(m^1)) \tag{4.9}$$

where $S(x(m)) = 1$ means that the fault at x(m) is detected by the simulated test pattern, otherwise, if $S(x(m)) = 0$, the fault is not detected. Since all the presented formulas are Boolean, the algorithms can be applied by tracing the nodes of the SSBDDs can be applied in parallel for many test patterns, each of them represented by one bit of the computer word. The number of operations needed for each node of SSBDD can calculate the cost of simulation. For example, the cost of logic simulation is four operations per node, and the cost of fault simulation is seven operations per node. Hence, to fault simulate the SSBDD in Fig.4.3 which includes nine nodes ,e.q. 9 * 7 = 63 (operations). Example of using the algorithms can be found in [68, 70]. Using SSBDDs instead of the gate-level circuit allows increasing both, the simulation speed for calculating the values of signals in the network of FFRs, and the fault reasoning, since only the collapsed fault set represented by nodes of SSBDDs is processed. This explains the efficiency of the method demonstrated in [68].

## 4.3 MIXED LEVEL FAULT SIMULATION WITH SSBDDS

Recently Shared SSBDDs ($S^3$BDD) as a new type of BDDs were proposed to speed-up logic simulation in digital circuits [48, 49]. In the following I propose a two level implementation of the proposed method of critical path tracing, where as the objectives of higher level, the fan-out nodes of the network of FFRs are considered, and as the objectives of lower level, the fan-out branches and fan-out free primary inputs of the network of FFRs are considered. The processing of formulas (4.4) for calculation of detectability of faults at fan-out nodes is carried out on the higher level using SSBDDs as in Fig. 4.3, and for computing the detectability of faults at the inputs of FFRs, I will use the data calculated by gate-level logic simulation.

In order to fault simulate all gate-level faults I propose to use S$^3$BDDs which can be processed in a similar way as SSBDDs. In Fig. 4.4, an S$^3$BDD is presented for calculation of the detectability of the faults at the inputs of FFRs. Each entry $x'$ in S$^3$BDD corresponds to a node variable $x(m)$ in the SSBDD in Fig.4.3, and the path from the particular entry to the terminal node represents an AND-function of conditions needed for detectability of the input variable $x$ of the given FFR. For example, the path in Fig.4.4 from the entry $3'$ through the nodes $\neg 2_2$, $c$, $\neg 7$, $a$ and $d$ to the terminal node #1 corresponds to the detectability condition of detecting the faults at the $input3$ of the FFR in Fig.4.3. Also Fig.4.4 shows how the faults located inside an FFR can be r̈eachedb̈y fault simulation. It can be seen that S$^3$BDD graph also has gate-level nodes $a'$ and $d'$ as input terminals. This way the S$^3$BDD model covers all the gate-level faults. Please note that in the notation of S$^3$BDD graph the arrows are directed from input nodes to the output. This however can be a point of confusion, as sensitivity calculations would run the other direction - from output to inputs. This way redundant computations are avoided and sensitivity results are saved along the computation process if any fault point(input terminal node) is reached.

The set of these detectability AND-functions for all of the input variables of the given FFR can be easily created from the gate-level structure of the FFR. To combine them in a form of S$^3$BDD like in Fig.4.4 the algorithm of optimized S$^3$BDD synthesis developed in [48] can be used.

The cost $C$ of fault simulation using S$^3$BDDs can be calculated in terms of the number of operations needed. For the gates with more than 2 inputs the synthesis process would make a 2-input gate equivalent. This way the cost can be computed using formula (4.10), where $i$ represents maximum number of inputs per gate in the given FFR and $j_n$ represents the number of gates with $n$ number of inputs, except the terminal gate at the output of the FFR. The internal sum is used to compute the number of operations needed for all the gates with the same number of inputs. And the second sum computes the overall amount of operations required for given S$^3$BDD by adding up the results for gates with different number of inputs. The number of operations for terminal output gate is represented by $N_t$, which concludes the formula.

$$C = N_t + \sum_{n=2}^{i} \sum_{1}^{j_n} 2(n-1) \tag{4.10}$$

For the S$^3$BDD model in Fig.4.4 I have $C = 12 + 3 = 15$, which is four times less than 63 operations needed for simulation of the SSBDD in Fig. 4.3. The calculation of the cost of SSBDD fault simulation is discussed in Section 4.2.3. Although the cost depends on the number of inputs of the gates inside the FFR it can be seen from Eq.4.10 the dependency is linear. It needs to be considered that if there are gates with many inputs inside an FFR, the FFR itself must have a big amount of inputs, which also drives the cost of SSBDD simulation higher. It guarantees the quicker execution of fault simulation using S$^3$BDDs with better fault resolution.

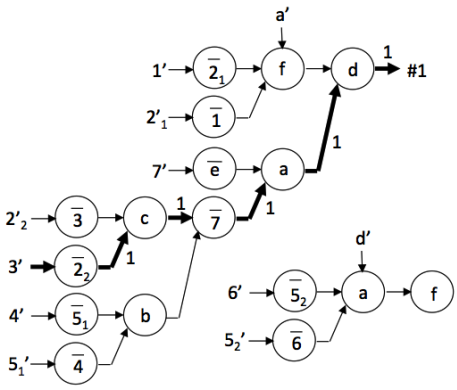



**Figure 4.4** – Direct fault simulation using S$^3$BDDs.

Consider, as an example, the mixed level work share in the computing processes of the level 2 in Table 4.1 between SSBDD and S$^3$BDD models. These processes handle the critical path tracing over the nested configuration of three fan-out re-convergence areas. In the process

$$(z_1, z_3) = R_{z_1}, z_3((z_1, z_{22}), (z_1, z_{12}) \equiv 1), \tag{4.11}$$

$(z_1, z_{22})$ is computed at the low-level on the S$^3$BDD for the FFR with output $z_2$, whereas $R_{z_1}$,$z_3$ is computed at the higher level using the SSBDD of $z_3$ after the following updates of the node variable values: $z_2 2 = z_2 2 \oplus (z_1, z_2 2)$, and $z_1 2 = \neg z_1 2$. On the other hand, in the process

$$D_{z_1} = ((z_1, z_3) \wedge D_{z_{31}}) \vee R_{z_1}, y5((z_1, z_3), (z_1, z_{13}) \equiv 1), \tag{4.12}$$

$D_{z_{31}}$ is computed at the low-level on the S$^3$BDD for the FFR with output $y_4$, whereas $R_{z_1}$,$y_5$ is computed at the higher level using the SSBDD of $y_5$ after the following updates: $z_{32} = z_{32} \oplus (z_1, z_3)$, and $z_{13} = \neg z_1$.

Additional side effect of the mixed-level fault reasoning is the increase of the accuracy in reporting the detected faults. Using the information about the gate-level structure of FFRs, allows specifying the detected faults inside the FFRs. For example, the entries $a'$ and $d'$ in the S$^3$BDD in Fig.4.4 are introduced to mark the sub-graphs for calculating the detectability of internal gate-level faults at the nodes $a$ and $d$, respectively, inside the FFR, presented in Fig.4.3. Similar entries can be added in Fig.4.4 for other internal nodes $b$, $c$, $e$, and $f$ in the same FFR.

The speed-up in mixed-level fault reasoning and the increasing accuracy of detected fault reporting is accompanied with additional time cost needed for logic simulation of FFRs at the gate-level. However, when comparing the total times for logic simulation and fault simulation this payload increase will be negligible.

## 4.4 MULTICORE FAULT SIMULATION USING SSBDDS

The multicore approach is based on the representation of the SSBDD and topological models of the circuit in leveled order. This is followed by level organized execution of fault simulation using OpenCL framework [23].

### 4.4.1 *Representation of levels*

The partitioning of the circuit into levels for concurrent execution have already been used before [4, 24, 75]. The level $i$ gate is defined in [4] as gate, which has primary inputs of the circuit and/or outputs of level $k$ gates as its inputs, such that $k < i$. However in [75] the definition is slightly different, stating that level of a gate represents its distance in gates from primary inputs (PI's) of the circuit. This definition is more strict in the sense that one of the inputs of the level $i$ gate, must originate from the level $i-1$, if $i \neq 0$. This difference however is crucial for parallelization, because the use of the first definition could potentially result in bigger number of levels with fewer gates in them. Because levels should be evaluated sequentially - this could decrease the amount of parallelism dramatically. I would stick to the second definition and rephrase it for the purpose of simulation on macro-level, using FFRs instead of gates.

SSBDD model describes the circuit as a set of primary inputs, graphs, representing FFRs and primary outputs. Here and throughout the chapter I would use the word *variable* to indicate these elements of the circuit. It means that FFRs, as well as primary inputs and outputs form the set of variables. The level of variable is its distance in variables from PI's. In other words, the level $i$ variable should have at least one of its inputs originating from level $i-1$ variable, if $i \neq 0$.

In SSBDD model, the variables are numbered in serial fashion starting at primary inputs and finishing at primary outputs. Variables are serialized such that each input of variable $i$ is the output of variable $k$, where $k < i$. This is very similar to the first definition of levels from [4]. As the OpenCL framework is used for parallel execution there is a requirement to able to run the same function with multiple data. Hence it is necessary to define regions of variables, belonging to the same level, as sub-array. Only variables of particular level must be included into sub-array. If variable $x$ belongs to level $i$, then level $i$ should be represented as a continuous sequence of variables starting from variable $x$ to variable $y$, such that every variable $z$ ($x \leq z < y$) belongs to level $i$ and variable $y$ belongs to level $i+1$. This is why it is necessary, to reorder the variables according to the definition of levels stated before. Note that this operation is only required once and does not belong to fault simulation process. The reordered SSBDD model can be saved as a file and used later for simulation, without a need to repeat this step. It should also be mentioned that the reordered model is identical to the original model in terms of circuit representation.

### 4.4.2 *Fault simulation process*

OpenCL framework requires single program for all the parallel devices, which would manipulate on different data. Such program is called a kernel. It is executed on all available devices in parallel for all variables inside a single level. The best way to provide the data for kernel is an array. During fault model preparation the variable indexes are placed into an array according to their levels. The kernel only requires knowing the offset of the level inside the array of variable indexes and the size of this level. Host CPU schedules the kernel executions level by level into the OpenCL execution queue. The execution in the queue is in order, so that OpenCL driver handles the synchronization between consecutive kernel executions. This ensures that all variables of the current level have been computed, before moving to the next level. The computation itself is a sequence of functions from the topological model, prebuild before the fault simulation process. The functions are used to compute the sensitivity of primary outputs to the change in value at the output of the particular variable.

## 4.5 EXPERIMENTAL RESULTS

The experiments were carried out on IBM System x3500 M3 7380 Server (2x 6-core Xeon E5690 running at 3,47Ghz with hyper-threading) using 64-bit Novell SuSe Linux Enterprise Server 11 x86_64. This system has 12 physical CPU cores, 12 virtual hyper-threading cores and 96Gb of RAM. Simulation times were calculated for the sets of 10000 random test patterns. The circuits from three benchmark suites ISCAS'85, ISCAS'89, ITC'99 were simulated. The same circuits as in [70] were chosen in order to compare the results.

Table 4.2 shows the results of the PECPT execution time $T'_p$ in comparison to PPECPT $T_{PPECPT}$ [69]. Along with execution time there are two speedup values I compute for every benchmark. These are $S_p$ and $S_c$. Both include single CPU (non-parallel) computation time of fault model $T_{tpl}$ and fault-free simulation $T_{ffs}$ of the circuit. Along with these $S_p$ uses parallel execution time $T_p$ for its computation and $S_c$ uses pure parallel computation time $T_c$, discussed later. The equations for speedup values $S_p$ and $S_c$ are as following:

$$S_p = \frac{T_{PPECPT}}{T_{tpl} + T_{ffs} + T_p} = \frac{T_{PPECPT}}{T'_p}$$
$$S_c = \frac{T_{PPECPT}}{T_{tpl} + T_{ffs} + T_c} = \frac{T_{PPECPT}}{T'_c}$$

$S_c$ can be though as topmost ideal case of speedup by PECPT algorithm. It can be seen from the results that smaller circuits achieve less speedup requiring less parallel hardware. On the other hand bigger circuits take advantage of higher number of processors. Overhead ratio $R$, for the case of maximum acceleration, is also brought in the table to see the concurrency overhead for different circuits. During the discussion I show the reasons behind the lower speedup of smaller

**Table 4.2** – Execution times of PPECPT and PECPT.

| Circuit | $T_{PPECPT}$, s | Concurrent overhead | | | | Pure computation | | |
|---|---|---|---|---|---|---|---|---|
| | | $T'_p$, s | $S_p$ | $S_p$ #cpu | R | $T'_c$ s | $S_c$ | $S_c$ #cpu |
| c1908 | 0,0568 | 0,0846 | 0,67 | 6 | 2,86 | 0,0330 | 1,72 | 5 |
| c2670 | 0,0405 | 0,0873 | 0,46 | 4 | 6,52 | 0,0334 | 1,21 | 6 |
| c3540 | 0,1830 | 0,1315 | 1,39 | 8 | 1,81 | 0,0754 | 2,43 | 7 |
| c5315 | 0,0849 | 0,0922 | 0,92 | 4 | 3,05 | 0,0487 | 1,74 | 5 |
| c6288 | 1,4610 | 0,6211 | 2,35 | 6 | 1,61 | 0,3883 | 3,76 | 8 |
| c7552 | 0,1545 | 0,1187 | 1,30 | 6 | 1,94 | 0,0718 | 2,15 | 6 |
| s13207 | 0,1798 | 0,1332 | 1,35 | 5 | 5,05 | 0,0857 | 2,10 | 10 |
| s15850 | 0,4714 | 0,2107 | 2,24 | 8 | 2,34 | 0,1370 | 3,44 | 7 |
| s35932 | 0,2554 | 0,1739 | 1,47 | 10 | 1,95 | 0,1381 | 1,85 | 12 |
| s38417 | 0,7453 | 0,2427 | 3,07 | 12 | 1,95 | 0,1869 | 3,99 | 12 |
| s38584 | 0,5945 | 0,2492 | 2,39 | 9 | 2,43 | 0,1791 | 3,32 | 12 |
| b14 | 2,7742 | 0,8752 | 3,17 | 8 | 1,29 | 0,7300 | 3,80 | 9 |
| b15 | 5,0420 | 1,1771 | 4,28 | 10 | 1,49 | 0,9258 | 5,45 | 10 |
| b17 | 14,8550 | 2,4053 | 6,18 | 20 | 1,29 | 2,1121 | 7,03 | 12 |
| b18 | 67,3279 | 7,1499 | 9,42 | 24 | 1,09 | 6,7738 | 9,94 | 24 |
| b19 | 147,6501 | 14,4685, | 10,20 | 24 | 1,03 | 14,0707 | 10,49 | 24 |

circuits and higher overhead ratio. It can be seen from the table that overhead ratio is getting close to one, with growth of the circuit size, getting speedup almost identical with ideal.

Speedup $S_p$ dependence on the number of processors is shown in Fig.4.5a (ISCAS'85), Fig.4.5b (ISCAS'89), Fig.4.5c (ITC'99). The fluctuation in speedup of some circuits can be explained by the fact that it is up to OpenCL runtime to decide which processors to use for execution. Because test system I used has virtual hyper-threading cores they can also be arbitrarily chosen for execution, which could influence the speed of execution in situations where less physical cores are used for computation, although the overall number of cores is bigger. For all the benchmarks it can be seen that after the limit of physical cores is reached the speedup is starting to decline or stays the same. On the ITC'99 benchmarks b18 and b19 it is slightly increasing, when more than 12 cores are used. This shows

that more speedup could be achieved on more powerful system with higher number of physical CPU cores.

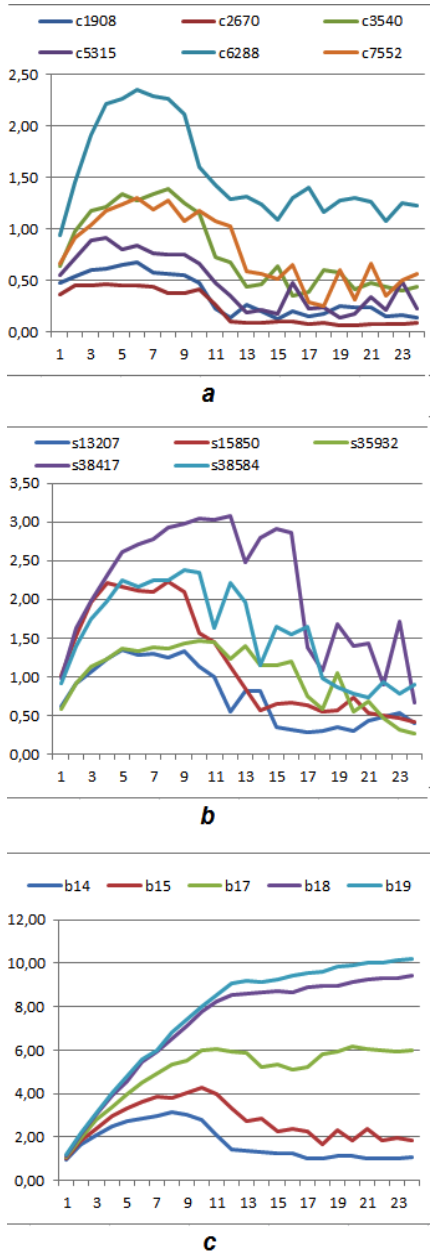



**Figure 4.5** – Speedup vs #CPU for PECPT. a). ISCAS'85 benchmarks, b). ISCAS'89 benchmarks, c). ITC'99.

### 4.5.1 *Discussion*

Parallel execution of PECPT is influenced by two factors. The first factor is the amount of parallelism available in the circuit. The second factor is the ratio of overall execution time to pure parallel computation time. I would discuss both of the factors below.

#### *Amount of parallelism*

The amount of parallelism available in the circuit can be expressed in average amount of computations per level. This number grows with the size of the circuit. This is why smaller circuits have less potential for speedup in proposed method, than bigger circuits, because they have fewer amounts of computations to be concurrently processed, so less hardware is needed for parallelization.

#### *Concurrency overhead*

Concurrent execution time $T_p$ can be divided into two parts: $T_p = T_o + T_c$. The first part is the time $T_o$, which I would call *concurrent overhead*. This is required to make a transition from "single thread"- to "multiple thread"-execution and back again. This time involves creation of multiple threads, allocating additional memory, synchronization at the end of computation and transition back to single thread. The second part is time $T_c$, which is pure *computation time* required by all threads to deliver a result. This time can be seen in Table 4.2 and can be treated as a lower possible bound for concurrent computation. The concurrent overhead $T_o$ depends on the amount of parallel hardware used and increases with number of CPUs. The computation time $T_c$ depends on the amount of computation required.

Amount of calculation for small circuits is small, which makes overall execution time $T_p$ large in comparison to computation time $T_c$. This can be expressed by overhead ratio $R = T_p/T_c$. I have carried out the set of experiments in order to prove and demonstrate the influence of this factor on the overall speedup in case of smaller circuits. ISCAS'85 benchmark circuits were simulated. The set of experiments consists of PECPT simulations were the amount of computation pro-

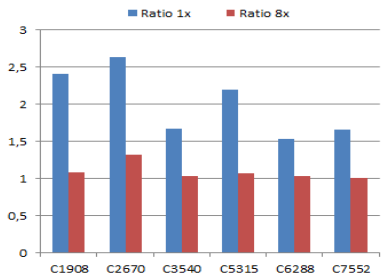



**Figure 4.6** – Overhead ratio R dependence on the amount of computation.

cessed per variable is increased 8 times. This way it can be seen how $T_p/T_c$ ratio changes with increasing amount of computation. Fig. 4.6 shows the ratio $R$ for 1x and 8x pattern bunches processed per each variable. It can clearly be seen that the amount of work brings the overall concurrent execution time $T_p$ very close to computation time $T_c$. This shows that if an amount of computation is sufficient the influence of $T_o$ becomes negligible even for a small circuit and $R \approx 1$. This is also seen from the results of ITC'99 benchmarks, which are considerably bigger then ISACS'85.

### 4.5.2 *Comparison*

I have compared PECPT to single processor simulators, which include FSIM, PPECPT and commercial simulators C1 and C2. The execution time of all the simulators was normalized using previous results from [70] and execution time of PPECPT from Table 4.2, because PECPT was executed on different hardware. The comparison is shown in Table 4.3.

**Table 4.3** – Execution time comparison.

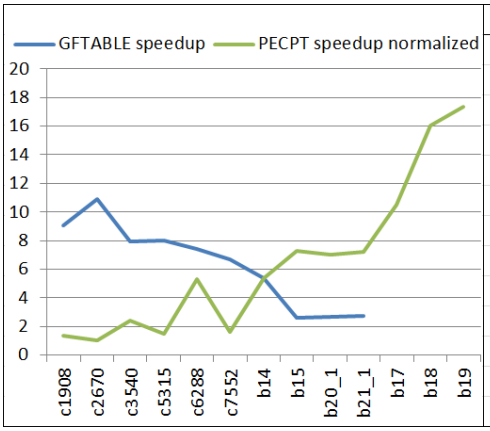
| | | #branches | | Simulation time,s | | | | |
|---|---|---|---|---|---|---|---|---|
| circuit | #fanouts | max | avg | fsim | c1 | c2 | *PPECPT* | *PECPT* |
| c2670 | 290 | 28 | 3,7 | 0,081 | 0,223 | 2,430 | 0,041 | 0,087 |
| c3540 | 356 | 22 | 4,5 | 0,407 | 1,505 | 8,745 | 0,183 | 0,132 |
| c5315 | 510 | 31 | 5 | 0,149 | 0,594 | 6,047 | 0,085 | 0,092 |
| c6288 | 1456 | 16 | 2,6 | 2,389 | 5,489 | 56,072 | 1,461 | 0,621 |
| c7552 | 812 | 72 | 4,1 | 0,348 | 1,043 | 11,332 | 0,155 | 0,119 |
| s13207 | 1224 | 37 | 3,7 | 0,225 | 0,503 | 6,291 | 0,180 | 0,133 |
| s15850 | 1518 | 34 | 3,6 | 0,943 | 2,112 | 19,379 | 0,471 | 0,211 |
| s35932 | 5295 | 1449 | 3,4 | 0,412 | 1,058 | 17,477 | 0,255 | 0,174 |
| s38417 | 4569 | 49 | 3,2 | 1,725 | 3,343 | 33,007 | 0,745 | 0,243 |
| s38584 | 3946 | 88 | 4,5 | 1,124 | 2,155 | 29,727 | 0,595 | 0,249 |
| *Average speedup* | | | | *3,786* | *8,748* | *92,460* | *2,024* | *1,000* |
| b14 | 2409 | 82 | 4,8 | n/a | 9,413 | n/a | 2,774 | 0,875 |
| b15 | 2353 | 95 | 4,8 | n/a | 7,411 | n/a | 5,042 | 1,177 |
| b17 | 8145 | 149 | 4,8 | n/a | 22,340 | n/a | 14,855 | 2,405 |
| *Average speedup* | | | | *n/a* | *8,774* | *n/a* | *4,118* | *1,000* |

PECPT proves to be around 3.5 times quicker than FSIM and around two times - than PPECPT for relatively smaller ISCAS benchmarks. The speedup over commercially available simulators is more than eight times over C1 and two orders of magnitude over C2. When ITC'99 benchmark circuits are also taken into consideration the average speedup over PPECPT grows to 4.0 in average, which suggests that simulation of bigger circuits benefits more from proposed method. On the other hand PECPT shows the same 8,7 times performance increase over commercial simulator C1 when bigger circuits are added, which suggests that C1 also behaves better when circuit size increases.

### *Comparison to GFTABLE*

I have also compared PECPT speedup results to GPU based parallel fault simulator and fault table generator GFTABLE [25]. GFTABLE is pattern parallel simulator, which uses bit- and thread-level PP to boost the performance of uniprocessor simulator FSIM. The results from Table 4 in [70] were used to normalize PECPT speedup. Normalization is required because PECPT speedup is computed in relation to PPECPT, while GFTABLE speedup is computed in relation to FSIM. As there is no FSIM execution time provided for ITC'99 benchmarks, I have taken the average ratio of 1.7 reported in [70] to normalize PECPT results for those circuits. The performance of both methods can be seen in Fig.4.7.

Even for the circuits, which could fit into GPU memory slight decrease in performance of GFTABLE can be seen. Contrary the results of presented approach become better while circuit size increases. Both methods use shared memory systems for execution. The comparison suggests that pattern-parallelism on such systems is better for smaller circuits, while circuit-parallelism becomes more advantageous for bigger circuits. The reason could lie in memory bottleneck of shared-memory, which effect increases more rapidly for pattern-parallel systems with the growth of the circuit size.




| curcuit | # gates |
|---|---|
| c1908 | 1394 |
| c2670 | 2075 |
| c3540 | 2784 |
| c5315 | 4319 |
| c6288 | 4846 |
| c7552 | 5795 |
| b15 | 18248 |
| b14 | 19491 |
| b21_1 | 28017 |
| b20_1 | 28019 |
| b17 | 64711 |
| b18 | 222499 |
| b19 | 448502 |

**Figure 4.7** – Comparison of GFTABLE and PECPT.

*Bio-impedance benchmark results*

The effect of concurrency overhead has one more interesting impact on the performance of the PECPT. To illustrate it I have simulated all of the gate-level benchmark circuits presented in Chapter 2 using both PPECPT and PECPT. The results are presented in Table 4.4 and in Fig.4.8. The table shows the simulation time for both single-core PPECPT and multi-core PECPT. In all the cases the multi-core version is faster, which is obvious as all the circuits are big enough to provide required level of parallelism. However if the changes in simulation time of different circuits are also compared it can be seen that multi-core version executes more complex circuits faster than single-core algorithm. It is probably not that obviously seen from the Table 4.4, but can clearly be seen from the Fig.4.8. Here the simulation time of all the benchmarks is normalized to the simulation time of the circuit 8a. For example for the circuit 8bs the simulation time of the PPECPT is about 550% of the 8a, but only about 350% for PECPT. The reason lies in the concurrency overhead. On the one hand the fault simulation based on back-tracing is highly sensitive to the number of reconvergent fan-out, as they require special time-consuming computation routines to be executed. But on the other hand the multi-core version takes advantage of bigger computational load, significantly reducing concurrency overhead and achieving better performance. This results in smaller difference between execution speed of bigger and smaller circuits.

## 4.6 CHAPTER SUMMARY

In this chapter the following main results were achieved:

**Table 4.4** – Simulation time for bio-impedance benchmark circuits using PPECPT and PECPT.

| design | PECPT, ms | PPECPT, ms | Overhead % |
|---|---|---|---|
| 8a | 2,32214 | 12,9366 | 25 |
| 8b | 2,1601 | 10,7997 | 24 |
| 8be | 4,97358 | 40,8165 | 12 |
| 8bk | 2,04 | 14,4935 | 25 |
| 8bs | 8,04758 | 71,7319 | 9 |
| 8c | 2,78738 | 15,1528 | 22 |
| 8d | 2,65016 | 19,9979 | 19 |
| 8de | 5,33164 | 50,757 | 9 |

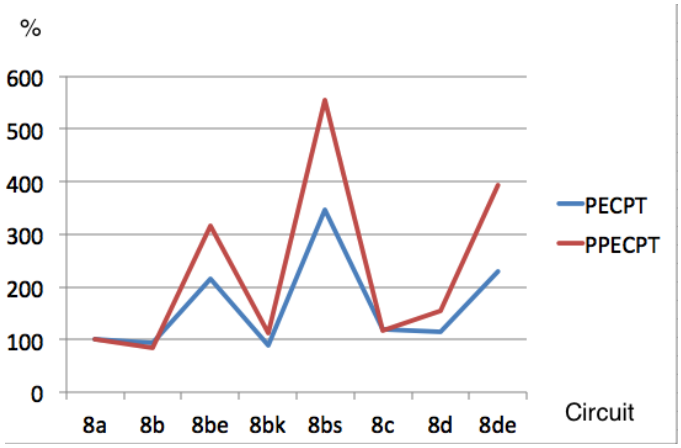



**Figure 4.8** – Changes in PPECPT and PECPT simulation times for bio-impedance benchmark circuits.

- A novel two-level method of critical path tracing was developed, which combines the FFR level and gate-level fault simulation and allows to increase the accuracy in reporting of detected faults.
- A novel multi-core exact critical path tracing based fault simulation method was developed which combines parallelism in three dimensions and allows to improve the speed of simulation in order of magnitude compared to the state-of-the-art commercial simulators.

The multicore parallelism is achieved by exploiting circuit-processing concurrency. The parallelization in fault simulation is carried out simultaneously in three dimensions: pattern parallelism, fault parallelism and computing model parallelism, where the pattern- and fault-parallelism are utilized using each single CPU core, while computing model parallelism is achieved using multiple CPUs.

Experiments showed that the average speed-up compared to the best uniprocessor based simulators is around 4.5 times. The method is well scaling, the speedup of the method grows with the size of the circuit, opposite to the pattern-parallel simulation method, which is more beneficial for smaller circuits. The reason lies in the memory bottleneck of shared-memory systems, which increases more rapidly for pattern-parallel systems with the growth of the circuit size. Comparison to uniprocessor fault simulators shows order of magnitude average speed-up over available state-of-the-art commercial simulators.

# 5

# COMBINATIONAL FAULT SIMULATION ENVIRONMENT FOR SEQUENTIAL CIRCUITS

This chapter is based on the publication "Combinational Fault Simulation in Sequential Circuits" (see Appendix E).

In this chapter a very fast fault simulation method is proposed for sequential circuits which is based on exact parallel critical path tracing developed for combinational circuits. To convert the sequential problem of fault simulation into the combinational one a set of MISRs is introduced into the circuit to improve its observability. The role of these MISRs is to monitor signals on the global feedback loops and on selected fan-out stems. The feasibility and correctness of the method is shown, and the experimental results, which demonstrate the speed-up achieved by the method, are presented and discussed.

The contribution of the author includes the development of the method as well as planning and running the experiments.

The rest of the chapter is organized as following. Section 5.2 describes how combinational fault simulation can be generalized for the case of sequential circuits. In Section 5.3 I describe experimental results and Section 5.4 concludes the chapter.

## 5.1 OVERVIEW

As it was already mentioned in previous chapter fault simulation is one of the most important tasks in digital circuit design and test. Therefore it is not a surprise that accelerating it would have a strong impact to a number of applications. It is extremely necessary to speed up sequential fault simulation, as most of the digital circuits nowadays are sequential.

I start with the overview of the fault simulation for combinational circuits, as it forms a base for method proposed in this chapter. Many different methods have been developed for fault simulation in combinational circuits based on the concept of parallel pattern single fault propagation (PPSFP) [77]. Another trend is based on the fault reasoning (deductive [7], concurrent [74] and differential simulation [16]) used to be very powerful, since these methods allow to collect all detectable faults by a single run of the given test pattern. What they cannot do, is to produce reasoning for many test patterns in parallel. As it was already mentioned in Section 4.2.2 this drawback was removed in [68] by introducing a

novel concept of Parallel Pattern Exact Critical Path Tracing (PPECPT) which can be applied efficiently for multiple patterns inside FFRs, as well as beyond them.

Unfortunately, for sequential circuits the parallelism exploited in combinational fault simulation and fault reasoning is not possible, because of the sequential (time related) dependence of signals in the circuit. The possible available solutions include fault simulation algorithms described in Section 4.1.

Although efficient all those methods iteratively resimulate the circuit either for all test patterns or non-dropped faults. In order to cover all the faults in one run I propose to modify the given circuit to improve its transparency (observability). The traditional way to do this is to use the scan-path concept [62] which converts the sequential problem of fault simulation to the combinational one. However, the use of scan-chains has proven to be often inadequate due to increasing the cost in terms of additional hardware and increased testing time [12], excessive power dissipation during test [78] and leading to yield loss because of over-testing [15].

In the following it will be shown that a sequential circuit can still be fault simulated as a combinational one when to improve its observability by inserting a set of Multiple Input Signature Registers (MISR), for monitoring of a selected subset of test points in the circuit. Two rules for selecting these test points to include MISRs are introduced and discussed. It is also shown how the test sequence can be mapped into a set of independent local test sequences which can be simulated in parallel similarly to the case of combinational circuits.

The target of this chapter is to provide a method of fault simulation in a modified circuit with a dramatic speed-up compared to the traditional non-parallel fault simulation of sequential circuits. Only the class of stuck-at-faults (SAF) is considered here. However, as shown in [70], the results can be extended to other fault classes like conditional SAF, transition delays, and X-faults.

## 5.2 MODIFICATION OF SEQUENTIAL CIRCUIT

The substantial problem of fault simulation in sequential circuits lies in the fact that the same fault can influence a particular component in different time frames. This fact excludes the possibility of exploiting the powerful critical path tracing based method, explained in the Section 4.2.2 of Chapter Chapter 4, for fault simulation in combinational circuits. The reason is in the exponential explosion of the number of nested and intersected re-converging fan-out regions over different time-frames. However, this problem as will be shown can be neglected if there will be a possibility to detect the fault in the first occasion when it has propagated up to the component.

There are two reasons why a fault can be propagated to the same component during different time frames: because of the global feedback which includes this component, and because of a re-convergent fan-out where the fault may propagate from the fan-out stem to the converging point by different number of clocks. If a MISR will be inserted into these "problem causing" test points, the fault can be captured always at the first occasion it influences on the component. The detection of the fault is fixed, and its impact in the future can be ignored. Note,

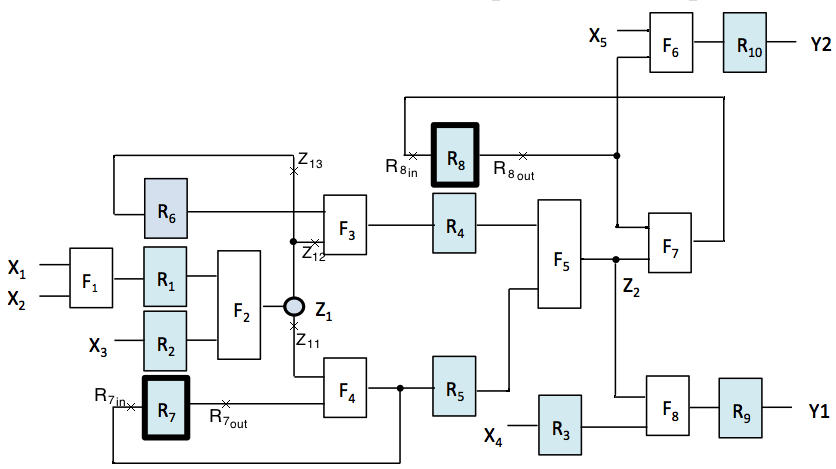



**Figure 5.1** – Sequential circuit.

only the problem of fault detection (for measuring the fault coverage) is considered here, and not the task of creating fault tables to be used for fault diagnosis purposes.

From above, two rules result for improving the observability of the sequential circuit:

- RULE 1: Insert a MISR to all registers (and only to them) which are included into a global feedback. Inserting a MISR is equivalent to cutting the feedback loop (in a sense to ignore the further fault propagation).
- RULE 2: Insert a MISR into all fan-out stems which have at least a single converging point, so that a fault may propagate from the fan-out stem to this point by different number of clocks.

Consider a sequential circuit in Fig.5.1 which consists of 9 registers (latches) $R_1 - R_9$, and 8 combinational sub-circuits $F_1 - F_9$. The circuit has 5 inputs and 2 outputs.

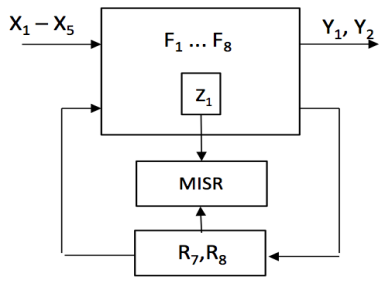



**Figure 5.2** – Sequential circuit with MISR.

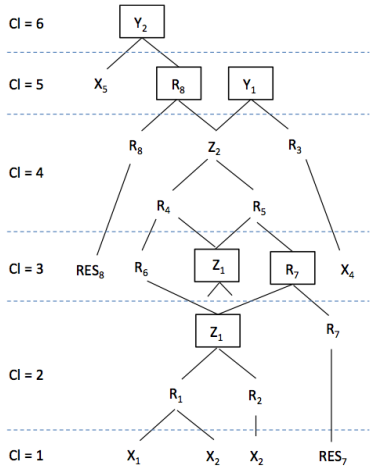



**Figure 5.3** – Simulation cycle of a single independent test.

In the circuit in Fig.5.1, two registers $R_7$ and $R_8$ are included into a global feedback loop, and hence, according to RULE 1, they must be furnished by MISR. On the other hand, there is a fan-out stem $Z_1$ which has two branching paths which re-converge in $F_3$. The first path represents a direct connection, and the second one is a path via register $R_6$, where the possible faulty signal needs for propagating from $Z_1$ to $F_3$ additional clock. Hence, according to RULE 2, the node $Z_1$ must be monitored by MISR. The modified circuit is presented in Fig.5.2.

In Fig.5.3, a simulation cycle of a single independent test sequence with lengths of 6 clocks is shown where by rectangles the 5 observation points are denoted. In this simulation cycle I can extract 5 functions (the upper indexes denote the delay in clock cycles between the moments when the values of argument signals and the function signal were fixed, respectively):

$$\begin{aligned} Z_1 &= f_{Z_1}(X_1^{-1}, X_2^{-1}, X_3^{-1}) \\ R_7 &= f_{R_7}(R_7^{-1}, Z_1^{-1}) \\ R_8 &= f_{R_8}(R_8^{-1}, R_7^{-2}, Z_1^{-2}, Z_1^{-3}) \\ Y_1 &= f_{Y_1}(R_7^{-2}, Z_1^{-2}, Z_1^{-3}, X_4^{-2}) \\ Y_2 &= f_{Y_2}(X_5^{-1}, R_8^{-1}) \end{aligned} \tag{5.1}$$

Since the arguments of these functions are either primary inputs of the circuit or the nodes supported by MISR, the set of functions (3) can be regarded as a model of 5 interconnected combinational circuits, which can be fault simulated independently.

Table 5.1 represents two (shifted in one clock cycle) input sequences of the two test segments $T_i$ and $T_{i+1}$, and the related output sequences captured by MISR

in the test points $Z_1$, $R_7$, $R_8$, and directly at outputs $Y_1$ and $Y_2$, which can be as well fed into MISR. The table represents the simulation order of the functions (3). Because the RULES 1 and 2 are satisfied in the modified circuit in Fig.5.2, the input sequences of $T_i$ and $T_{i+1}$,can be regarded as independent test patterns, spread merely over different time frames. In this way, a full test sequence applied to the circuit in Fig.5.2 can be split into a set of independent test segments, all shifted by one clock one after another. Since the test segments can be treated as a set of independent test patterns, they can be fault simulated by PPECPT in parallel as in case of combinational circuits.

Fig. 5.4 shows the equivalent combinational schematic for the circuit from Fig.5.1 constructed using Rules 1 and 2. The registers are left here for the illustrational purpose only for better understanding of the timing of different signal values. In a real case they have to be exchanged by wires. It can be seen from the Fig.5.4 that according to Rule 1 the registers R7 and R8 are broken up, such that input of the particular register becomes auxiliary output and the output of the register becomes auxiliary input. Also according to Rule 2 the fan-out stem $Z_1$ was broken up into three auxiliary inputs, because it has a converging point at register $R4$, where line $Z_{13}$ has register $R6$ and line $Z_{12}$ doesn't have any register. It means the number of clock cycles is different for a signal to arrive along these paths to a converging point, which qualify $Z1$ to Rule 2.

As can be seen from the Fig.5.4 the node values throughout the circuit are taken from different clock periods. Let $t$ be a value of the signal node at the current clock period. Then $t-1$ is the signal value of the particular node at the previous time period and so on. Going from outputs to inputs the clock period is decremented once the register is crossed. This way the test sequence can be derived, which contains the values of every node taken at different clock periods. We can then fed this sequence into the combinational fault simulator along with the equivalent combinational circuit, where registers become wires.

**Table 5.1** – A test sequence for circuit in Fig.5.2

| Cl | Input sequences | | Output sequences | |
|---|---|---|---|---|
| | Test $T_i$ | Test $T_{i+1}$ | Test $T_i$ | Test $T_{i+1}$ |
| 1 | $X_1^1$, $X_2^1$, $X_3^1$, $RES_7^1$ | | | |
| 2 | $X_1^2$, $X_2^2$, $X_3^2$ | $X_1^2$, $X_2^2$, $X_3^2$, $RES_7^2$ | $Z_1^2$ | |
| 3 | $R_8^3$, $X_4^3$ | $X_1^3$, $X_2^3$, $X_3^3$ | $R_7^3$, $Z_1^3$ | $Z_1^3$ |
| 4 | | $R_8^4$, $X_4^4$ | | $R_7^4$, $Z_1^4$ |
| 5 | $X_5^5$ | | $R_8^5$, $Y_1^5$ | |
| 6 | | $X_5^6$ | $Y_2^6$ | $R_8^6$, $Y_1^6$ |
| 7 | | | | $Y_2^7$ |

## 5.3 EXPERIMENTAL DATA

As experimental results in Table 5.2 in our laboratory we have compared the speed of SAF simulation in sequential circuits (where all the latches are fed into MISR) by the PPECPT method with different known fault simulators for combinational circuits: FSIM [40], and two state-of-the-art commercial simulators C1 and C2 from major CAD vendors. Simulation times were calculated for 10000 patterns. Experiments were run on a 1.5GHz Ultra SPARC IV+ workstation using SunOS 5.10.

Although the method is not yet implemented I can still show the efficiency it provides in comparison to traditional sequential fault simulation. For comparison I've picked up serial fault simulation, which simulates faults one by one for every pattern. Although slow this method is good for comparative analysis, as it provides the lowest bound in sequential fault simulation. It is easy to obtain the approximate time for such simulation by running traditional logic simulation for all the test vectors. Then the time obtained have to be multiplied by number of stuck-at faults in the circuit. I've simulated VHDL representations of the benchmark circuits presented in Chapter 2, hence achieving the fastest runtime for serial fault simulation. On the other hand, I took single-core pattern parallel critical path tracing fault simulation method along with full-scan combinational equivalents of the sequential benchmark circuits used for comparison. ]

I investigated the feasibility of the proposed fault simulation method for calculating the fault coverage of the at-speed functional self-test developed for these processors. The results of fault simulation for the whole family of 8 processors (column 1) are presented in Table 5.3 where LS denotes the behavior level logic simulation time, FS denotes the LS multiplied by the number of faults to be simulated one by one, and the PPECPT shows the simulation time needed for the

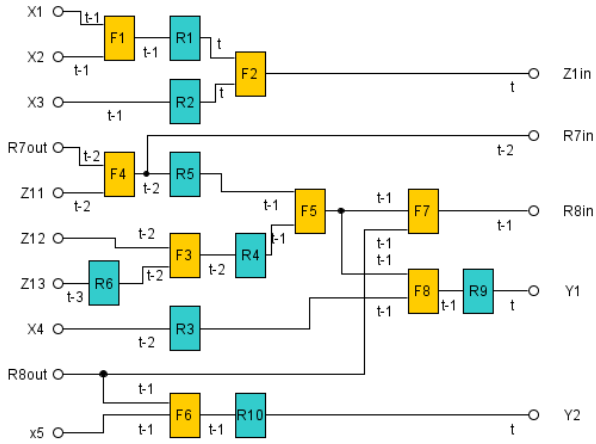



**Figure 5.4** – Combinational circuit equivalent for sequential circuit.

proposed method. The experiments showed that the gain achieved by using the proposed method could be up to 2-3 orders of magnitude. For this advantage the price have to be paid at the cost of added set of MISRs, which however is comparable to the cost of scan-path or can be even less.

In order to correctly interpret the results I highlight the major differences of methods used in experiment from methods used in real test cases. The serial fault simulation is typically run on gate-level, to simulate the gate-level stuck-at fault effect. In case of experiment the behavior level simulation is used, which is faster than gate-level, as it doesn't compute signal values at every gate. So the speed of real serial fault simulation is definitely slower, than it is shown during the experiment in Table 5.3. Also I do not use fault dropping for the sake of comparison, as PPECPT method doesn't use it either. The fault dropping can speed up the serial fault simulation making it faster than it is shown in Table 5.3.

As can be seen from the Fig. 2.4 in Chapter 2 the performance of PPECPT method depends on the amount of reconvergency in the circuit. At the same time we use the full-scan equivalent circuits in the experiments, which make the number of reconvergent fan-outs smaller. When using the method proposed in this chapter the amount of observability points should become smaller, increasing the amount of reconvergency in the circuit. Hence the PPECPT method may run slower than it is shown in Table 5.3. Taking all mentioned above in account the right way to interpret these experimental results would be as the upper limit of real case scenario.

**Table 5.2** – Comparison of PPECPT with other fault simulation methods for circuits with full scan-path.

| Circuit | Number of gates | SAF simulation time, s | | | |
|---|---|---|---|---|---|
| | | Fsim | C1 | C2 | PPECPT |
| c3540 | 2784 | 2.0 | 7.4 | 43 | 0.9 |
| c5315 | 4319 | 1.4 | 5.6 | 57 | 0.8 |
| c6288 | 4846 | 12.1 | 27.8 | 284 | 7.4 |
| s15850 | 14841 | 5.4 | 12.1 | 111 | 2.7 |
| s38417 | 34831 | 16.2 | 31.4 | 310 | 7.0 |
| s38584 | 36173 | 12.1 | 23.2 | 320 | 6.4 |
| b14 | 19491 | N/A | 49.2 | N/A | 14.5 |
| b15 | 18248 | N/A | 39.1 | N/A | 26.6 |
| b17 | 64711 | N/A | 117 | N/A | 77.8 |
| Average speed gain | | 2.0 | 4.3 | 45 | 1 |

**Table 5.3** – Comparison of the proposed method with single fault simulation in sequential circuits.

| Circuits | Number of faults | SAF simulation time, s | | | Gain |
|---|---|---|---|---|---|
| | | LS | FS | PPECPT | |
| 8a | 112034 | 0.155 | 17365 | 30.0 | 579 |
| 8b | 83940 | 0.152 | 12759 | 24.7 | 517 |
| 8be | 99330 | 0.168 | 16687 | 62.1 | 269 |
| 8bk | 86878 | 0.159 | 13814 | 25.2 | 548 |
| 8bs | 100820 | 0.154 | 15526 | 173.4 | 90 |
| 8c | 122386 | 0.159 | 19459 | 35.9 | 542 |
| 8d | 123012 | 0.161 | 19804 | 35.5 | 558 |
| 8de | 136876 | 0.164 | 22447 | 81.3 | 276 |

## 5.4 CHAPTER SUMMARY

In this chapter the following main results were achieved:

- A new method of design-for testability was proposed, which allows to convert a sequential circuit into an equivalent iterative combinational array, by inserting a small set of observation points into circuit to be connected with MISR.
- The parallel critical path tracing method for combinational circuits was reused for sequential circuits, which allowed a dramatic speed-up of fault simulation compared to the traditional single fault simulation for sequential circuits.

The high speed is achieved thanks to removing the problem of sequential dependence of simulated signals in different time frames by improving observability of the circuit by inserting a set of MISRs at selected test points. Therefore the essential requirement for the method to work is to modify the sequential circuit according to rules described in Section 5.2 and to create an equivalent combinational circuit with auxiliary inputs and outputs. The equivalent circuit can then be simulated using fast combinational fault simulator without loosing any accuracy in the fault coverage estimation.

Apart from a full-scan design, only a fraction of registers with global feedback need to be monitored. Additional MISRs must also be added to reconverging fan-out stems where fan-out signals arrive to reconverging point with different delay. A pessimistic experimental setup shows dramatic speed-up of fault simulation, compared to the traditional non-parallel fault simulation of sequential circuits.

# 6

# CONCLUSIONS

This chapter draws the overall conclusions of the thesis and provides some future possible directions to extend the results presented in previous chapters. The new research results in the thesis can be classified into five groups covering: (1) the set of benchmark circuits with different amount of resource sharing and pipelined architectures, (2) the methodology and a set of tools targeting at-speed built-in self test of high-performance pipe-lined designs, (3) the evaluation of possibility to use analog signals as a test sequence for the digital circuits, (4) the implementation of the PPECPT algorithm for general-purpose multicore systems, (5) the methodology to use a combinational fault simulation for sequential circuits.

## 6.1 BENCHMARK SUITE

A benchmark suite was developed for evaluating the CAD tools in their efficiency and quality in designing dependable digital systems. Apart from all other existing benchmarks, all the circuits of this family perform the same function, but differ mainly in the amount of shared computing resources. This gives an excellent possibility for direct systematic characterization of CAD tools in terms of alternative design decisions, which is not provided by existing benchmark suites. The experiments show a correlation between the structural properties of circuits and their testability characteristics. It was shown that sharing of resources in designs, which leads to increasing number of fan-out reconvergencies, may reduce the test length, but on the other hand, will increase the time of test synthesis, reducing the quality of the test.

## 6.2 METHODS FOR TESTING

METHODOLOGY AND SET OF TOOLS FOR AT-SPEED TEST A new approach to self-testing of digital systems with pipe-lined architectures with capability to produce internal self-test sequences using their inherent functionalities was proposed. The added value is the higher test quality explained by on-line at-speed testing. The approach does not need to store high volume test data in the system memory. Additional hardware is as well not needed for on-line test pattern generation as in the case of traditional LBIST. The only needed ad-

ditions related to using MISR for monitoring the test responses. To minimize the needed additional MISR hardware overhead, an original algorithm for selecting test-points was developed. As the result of using only normal working sequences for test purposes, the dangers of over-testing and the related yield loss are removed. Also a novel evaluation environment was developed where the time consuming sequential fault simulation task can be transferred into a set of combinational fault simulation sub-tasks. Experiments demonstrated the gain in evaluation speed more than 580 times without losing any accuracy in fault coverage calculation.

EVALUATION OF ANALOG SIGNALS AS A TEST SEQUENCE The goals of the experiments were twofold: (1) to select the best type of input signal for testing purposes from a set of signals typically used for processing in the given Signal Analyzer, and (2) to compare the new method with traditional scan path based testing methods. The fault coverage achieved by the sine signal was 98.2%, which is nearly the same compared to the traditional scan-path pseudorandom (98.7%) and deterministic (98.7%) test approaches. The gain in testing time cost was 3-7 times compared to the deterministic and more than 2500 times compared to the pseudorandom single scan-path based approach.

## 6.3 METHODS FOR TEST QUALITY EVALUATION

MULTI-CORE PECPT A new method for multi-core execution of pattern parallel exact critical path tracing(PPECPT) based fault simulation was proposed and implemented. The parallelism is achieved by exploiting circuit-processing concurrency. The parallelization in fault simulation is carried out simultaneously in three dimensions: pattern parallelism, fault parallelism and computing model parallelism, where the pattern- and fault-parallelism are utilized using each single CPU core, while computing model parallelism is achieved using multiple CPUs. A novel mixed level technique for fault reasoning was proposed to speed up and to increase the accuracy of fault simulation, compared with previous methods. Experiments showed that the average speed-up compared to the best uniprocessor based simulators is around 4.5 times. Comparison to commercial uniprocessor fault simulators shows order of magnitude average speed-up of the algorithm. The method is well scaling, because its performance grows with the size of the circuit, opposite to the pattern-parallel simulation method GFTABLE, which seems to be more beneficial for smaller circuits.

COMBINATIONAL FAULT SIMULATION OF SEQUENTIAL CIRCUITS A novel approach for fault simulation in sequential circuits is proposed which allows to achieve dramatic speed-up in simulation time compared to the traditional single fault simulation in sequential circuits at reduced cost of additional hardware. Apart from a full-scan design, only a fraction of registers and fan-out points need to be monitored. The high speed is achieved thanks to removing the problem of sequential dependence of simulated signals in different time frames by

improving observability of the circuit by inserting a set of MISRs at selected test points. MISRs must be added to reconverging fan-out stems where fan-out signals arrive to reconverging point with different delay and registers with global feedback. A pessimistic experimental setup shows dramatic speed-up of fault simulation, compared to the traditional non-parallel fault simulation of sequential circuits.

## 6.4 FUTURE WORK

One of the possible directions for at-speed based functional BIST is to find the ways to further improve the fault coverage. Although it is contrary to avoiding of overtesting, some applications would definitely require achieving maximum fault coverage possible. Another direction lies in addition of other fault models to the methodology. For example delay faults can be computed using stuck-at fault simulation results as shown in [36], which is directly applicable to the current work. Finally the solid framework of tools can be developed to fully automate the proposed method for integration into industrial test processes. Logic simulation could be carried out by PPECPT along with fault simulation. This could possible remove the necessity to have VHDL models of the circuits and use an external logic simulator. Also MISR integration into the circuit can be automated.

The major issue of multi-core PPECPT algorithm is undefined dependency of its performance on the structure of the circuit. Particularly the metric to estimate the number of CPUs to be used for optimal performance is yet to be found. The structure of the circuit could be a possible place to extract such a metric. This would require examining the effect of different structural characteristics and possibly their combinations on the performance of PPECPT. Another direction is to utilize the power of Graphic Processing Units (GPUs) as an additional computational resource in order to achieve even more gain in performance where possible. Although the OpenCL framework provides direct support to use both GPU and CPU for joint computations the algorithm must be optimized in different manner for each of the platforms in order to achieve good performance on both of them simultaneously [61].

The next logical step for the combinational fault simulation of sequential circuits is the actual implementation of the method. This would require the analysis of the structure of the circuit to identify the points for observation. Also the generation of equivalent combinational circuit is required to make a process fully automated. The implementation could be used to better explore the potential of the method and also identify possible issues, which can arise when using it in practice.

Finally both multi-core PECPT and combinational fault simulation of sequential circuits can be integrated into at-speed functional BIST methodology. PECPT can be used directly to exchange PPECPT to provide a gain in speed for evaluation of fault coverage of combinational parts of the circuit. The combinational fault simulation of sequential circuits could be used to provide a speed up of sequential parts of the circuit when dealing with optimizations related to num-

ber of MISRs and circuit partitioning. Altogether it should dramatically improve the scalability of the methodology, which could become feasible to be used with large industrial designs.

# BIBLIOGRAPHY


[1] OpenRISC website. [WWW] http://openrisc.io/ (05.10.2015).

[2] M. Abramovici, P. R. Menon, and D. T. Miller. Critical path tracing - an alternative to fault simulation. In *The 20th ACM/IEEE Design Automation Conference (DAC 1987)*, volume 20, pages 2–5, 1987.

[3] M. F. AlShaibi and Ch. Kime. Mfbist: A bist method for random pattern resistant circuits. In *The IEEE International Test Conference (ITC 1996)*, pages 176–185, October 1996.

[4] M. B. Amin and B. Vinnakota. Data parallel fault simulation. In *The IEEE International Conference on Computer Design VLSI in Computers and Processors (ICCD 1995)*, pages 610–615, 1995.

[5] P. Annus, A. Kuusik, R. Land, O. Märtens, and A. Ronk. A digital multichannel bioimpedance analyser: Signal processing task and its solution. In *The IEEE Instrumentation and Measurement Technology Conference (IMTC 2006)*, Sorrento, Italy, April 2006.

[6] K. J. Antreich and M. H. Schulz. Accelerated fault simulation and fault grading in combinational circuits. In *The IEEE Transactions On Computer-Aided Design*, volume 6, pages 704–712, 1987.

[7] D. B. Armstrong. A deductive method for simulating faults in logic circuits. In *IEEE Transactions On Computers*, volume 21, pages 464–471, 1972.

[8] V.S. Bagad. *VLSI Design*. Technical Publications Pune, 1 edition, 2008.

[9] F. Baronti, R. Roncella, R. Saletti, P. D'Abramo, L. Di Piro, H. Fabian, and M. Giardi. The importance of at-speed scan testing: an industrial experience. In *The 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*, pages 672–675, Lubeck, August 2007.

[10] F. Brglez, D Bryan, and K. Kominski. Combinational profiles of sequential benchmark circuits. In *The IEEE International Symposium on Circuits and Systems (ISCAS 1989)*, pages 1929–1934, 1989.

[11] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits. In *The IEEE International Test Conference (ITC 1985)*, pages 785–794, 1985.

[12] L. Bushard, N. Chelstrom, S. Ferguson, and B. Keller. Dft of the cell processor and its impact on eda test software. In *The IEEE Asian Test Symposium (ATS 2006)*, pages 369–374, 2006.

[13] M Bushnell and V Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*, volume 17. Springer Verlag, 2002.

[14] M. Čepin. *Assessment of Power System Reliability*. Springer Verlag London, 1 edition, 2011.

[15] L. Chen, S. Ravi, A. Raghunathan, and S. Dey. A scalable software-based self-test methodology for programmable processors. In *The IEEE/ACM Design Automation Confernce (DAC 2003)*, pages 548–553, 2003.

[16] W. T. Cheng and M. L. Yu. Differential fault simulation: a fast method using minimal memory. In *The ACM/IEEE Design Automation Conference (DAC 1989)*, pages 424–428, 1989.

[17] F. Corno, M.S. Reorda, and G. Squillero. Rt-level itc'99 benchmarks and first atpg results. *The IEEE Design Test of Computers*, 17(3):44–53, July 2000.

[18] R. Dorsch and H j. Wunderlich. *Accumulator based deterministic BIST*. ITC D.C, Washington, 1998.

[19] P. Ellervee, P. Annus, and M. Min. High speed data preprocessing for bioimpedance measurements: Architectural exploration. In *The 27th IEEE NORCHIP Conference*, pages 1–4. IEEE, 2009.

[20] H. Esmaeilzadeh, E. Blem, R.S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *The ACM 38th Annual International Symposium on Computer Architecture*, pages 365–376. ACM, 2011.

[21] J. Gaisler. A portable and fault-tolerant microprocessor based on the sparc v8 architecture. In *The IEEE International Conference on Dependable Systems and Networks (DSN 2002).*, pages 409–415, 2002.

[22] D. Gizopoulos and et al. Systematic software-based self-test for pipelined processors. *The IEEE VLSI Systems Conference*, 16(11):1441–1453, November 2008.

[23] Khronos Group. OpenCL standard for parallel programming of heterogenious systems. [WWW] http://www.khronos.org/opencl (24.09.2015).

[24] K. Gulati and S. P. Khatri. Towards acceleration of fault simulation using graphics processing units. In *The 45th ACM/IEEE Design Automation Conference 2008 (DAC 2008)*, pages 822–827, 2008.

[25] K. Gulati and Khatri S. Fault table generation using graphics processing units. In *The IEEE International High Level Design Validation and Test Workshop (HLDVT 2009)*, pages 60–67, 2009.

[26] Kyunghwan Han and Soo-Young Lee. A parallel implementation of fault simulation on a cluster of workstations. In *The IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008)*, pages 1–8, 2008.

[27] S. Hellebrand, J. Rajski, S. Tarnick, B. Courtois, and S. Venkataraman. Built-in test for circuits with scan based on reseeding of multi-polynomial linear feedback shift registers. In *IEEE Transactions On Computers*, volume 44, pages 223–233, February 1995.

[28] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski. Logic bist for large industrial designs. In *The IEEE International Test Conference (ITC 1999)*, volume 358-367, 1999.

[29] P. D. Hortensius, R. D. McLeod, and B. W. Podaima. Cellular automata circuits for bist. *IBM Journal of Research and Development.*, 34(2.3):389–405, 1990.

[30] N. Ishiura, M. Ito, and S. Yajima. Dynamic two-dimensional parallel simulation technique for high-speed fault simulation on a vector processor. In *The IEEE Computer-Aided Design of Integrated Circuits and Systems Conference*, pages 868–875, 1990.

[31] H. j. Wunderlich and G. Kiefer. Bit flipping bist. In *The IEEE International Conference On Computer Aided Design (ICCAD 1996)*, pages 337–343, November 1996.

[32] G. Jervan, P. Eles, Z. Peng, R. Ubar, and M. Jenihhin. Hybrid bist time minimization for core-based systems with stumps architecture. In *The 18th Int Symposium on Defect and Fault Tolerance in VLSI Systems*, Cambridge, MA, USA,, November 2003.

[33] D. E. Knuth. *The art of computer programming*, volume 2. Addison-Wesley, 1981.

[34] M. Kochte, M. Schaal, H.-J. Wunderlich, and C.G. Zoellin. Efficient fault simulation on many-core processors. In *The ACM/IEEE Design Automation Conference (DAC 2010)*, Anaheim, California, USA, June 2010. ACM.

[35] B. Koenemann. Lfsr-coded test patterns for scan designs. In *The IEEE European Test Conference (ETC 1991)*, pages 237–242, March 1991.

[36] J. Kõusaar, R. Ubar, S. Devadze, and J. Raik. Critical path tracing based simulation of transition delay faults. In *The Euromicro Conference on Digital System Design (DSD 2014)*, pages 1–6, Verona, Italy, August 2014.

[37] D. Krishnaswamy, E. M. Rudnick, J. H. Patel, and P. Banerjee. Spitfire: scalable parallel algorithms for test set partitioned fault simulation. In *The IEEE VLSI Test Symposium*, pages 274–281, 1997.

[38] H. Kruus. *Optimization of BIST in Digital Systems.* PhD thesis, Tallinn University of Technology, Tallinn, 2011.

[39] H. Kruus, R. Ubar, P. Ellervee, M. Brik, M. Gorev, M. Kruus, E. Orasson, V. Pesonen, P. Annus, M. Min, and K. Meigas. A benchmark suite for evaluating the efficiency of test tools. In *The IEEE Baltic Electronics Conference*, Tallinn, October 2012.

[40] H.K. Lee and D.S. Ha. Soprano: An efficient automatic test pattern generator for stuck-open faults in cmos combinational circuits. In *The ACM/IEEE Design Automation Conference (DAC 1990)*, Orlando,FL, June 1990.

[41] H.K. Lee and D.S. Ha. An efficient, forward fault simulation algorithm based on the parallel pattern single fault propagation. In *The IEEE International Test Conference (ITC 1991)*, pages 946–955, 1991.

[42] M. Li and M.S. Hsiao. 3-d parallel fault simulation with gpgpu. In *The IEEE Computer-Aided Design of Integrated Circuits and Systems Conference*, volume 30. IEEE, October 2011.

[43] T. Mak, S. Krstic, K. t. Cheng, and L. c. Wang. New challenges in delay testing of nanometer, multi-gigahertz designs. *The IEEE Design & Test of Computers Conference*, 21(3):241–248, 2004.

[44] N. Mazurova, J. Smahtina, and R. Ubar. Hybrid functional bist for digital systems. In *The 9th IEEE Biennial Baltic Electronics Conference (BEC 2004)*, pages 205–208, Tallinn, October 2004.

[45] M. Min, P. Annus, R. Land, T. Paavle, E. Haldre, and R. Ruus. Bioimpedance monitoring of tissue transplants. *The IEEE Instrumentation and Measurement Technology Conference (IMTC 2007)*, pages 1–4, 2007.

[46] M. Min, R. Land, O. Märtens, T. Parve, and A. Ronk. A sampling multichannel bioimpedance analyzer for tissue monitoring. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 902–905, 2004.

[47] S. Minato. *BDDs and Applications for VLSI CAD*. Kluwer Academic Publishers, 1996.

[48] D. Mironov and R. Ubar. Lower bounds of the size of shared structurally synthesized bdds. In *IEEE 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 77–82, 04 2014.

[49] D. Mironov, R. Ubar, and J. Raik. Logic simulation and fault collapsing with shared structurally synthesized bdds. In *IEEE European Test Symposium (ETS 2014)*, pages 26–30, 5 2014.

[50] G.E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, April 1965.

[51] B. Nadeau-Dostie. *Design For At-Speed Test, Diagnosis and Measurement*. Kluwer Academic Publishers, 2002.

[52] T. Paavle, M. Min, and T. Parve. Using of chirp excitation for bioimpedance estimation: Theoretical aspects and modeling. In *The 11th International Biennial Baltic Electronics Conference (BEC 2008)*, pages 325–328, 2008.

[53] S. Parkes, P. Banerjee, and J. Patel. A parallel algorithm for fault simulation based on proofs. In *The IEEE International Conference on Computer Design:VLSI in Computers and Processors (ICCD 1995)*, pages 616–621, 1995.

[54] S. Patil and P. Banerjee. Performance trade-offs in a parallel test generation/-fault simulation environment. In *The IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems*, pages 1542–1558, 1991.

[55] S. Patil, P. Banerjee, and J. H. Patel. Parallel test generation for sequential circuits on general-purpose multiprocessors. In *The ACM/IEEE Design Automation Conference (DAC 1991)*, pages 155–159, 1991.

[56] J. Phelps, C. Johnson, C. Goodrich, and A. Kokrady. The importance of functional-like access for memory test. In *The IEEE International Test Conference (ITC 2008)*, page 1, Santa Clara, CA, October 2008.

[57] J. Raik and et al. Turbo tester manual. [WWW] www.pld.ttu.ee/tt/ (28.09.2015).

[58] J. Rajski and J. Tyszer. *Arithmetic BIST in embedded systems*. Prentice-Hall, N J, 1998.

[59] S. Saxena, C. Hess, H. Karbasi, A. Rossoni, S Tonello, P. McNamara, S. Lucherini, S. Minehane, C. Dolainsky, and M. Quarantelli. Variation in transistor performance and leakage in nanometer-scale technologies. In *The IEEE Transactions on Electron Devices*, volume 55, pages 131–144, January 2008.

[60] M. Shafique, S. Garg, J. Henkel, and D. Marculescu. The eda challenges in the dark silcon era. In *The ACM/EDAC/IEEE 51st Design Automation Conference (DAC 2014)*, pages 1–6, San Francisco, CA, June 2014.

[61] J. Shen, J. Fang, H. Sips, and A.L. Varbanescu. Performance traps in opencl for cpus. *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 38–45, February 2013.

[62] L. t. Wang, C. w. Wu, and X. Wen. *VLSI test principles and architectures*. Morgan Kaufmann, 2006.

[63] A. Thayse. *Boolean Calculus of Differences*. Springer Verlag, 1981.

[64] N. A. Touba and E. J. McCluskey. Bit-fixing in pseudorandom sequences for scan bist. In *The IEEE Transactions on CAD of IC and Systems*, volume 20, page 4, April 2001.

[65] R. Ubar. Test synthesis with alternative graphs. In *The IEEE Design and Test of Computers.*, pages 48–59, 1996.

[66] R. Ubar. Combining functional and structural approaches in test generation for digital systems. *Journal of Microelectronics and Reliability, Elsevier Science Ltd.*, 38(3):317–329, 1998.

[67] R. Ubar. Multi-valued simulation of digital circuits with structurally synthesized binary decesion diagrams. *Multiple Valued Logic Journal*, 4:141–157, 1998.

[68] R. Ubar, S. Devadze, J. Raik, and A. Jutman. Ultra fast parallel fault analysis on structural bdds. In *The 12th IEEE European Test Symposium (ETS 2007)*, pages 20–24, May 2007.

[69] R. Ubar, S. Devadze, J. Raik, and A. Jutman. Parallel fault backtracing for calculation of fault coverage. In *The 13th Asia and South Pacific Design Automation Conference (ASP-DAC 2008)*, pages 667–672, March 2008.

[70] R. Ubar, S. Devadze, J. Raik, and A. Jutman. Parallel x-fault simulation with critical path tracing technique. In *The IEEE Conference on Design, Automation & Test in Europe*, pages 1–6, March 2010.

[71] R. Ubar, V. Indus, O. Kalmend, and T. Evartson. Functional built-in self-test for processor cores in soc. In *The 30th IEEE NORCHIP Conference*, Copenhagen, Denmark, November 2012.

[72] R. Ubar, N. Mazurova, J. Smahtina, E. Orasson, and J. Raik. Hyfbist: Hybrid functional built-in self-test in microprogrammed data-paths of digital systems. In *The IEEE International Conferece MIXDES*, pages 497–502, Szczecin, June 2004.

[73] R. Ubar, T. Shchenova, G. Jervan, and Z. Peng. Energy minimization for hybrid bist in a system-on-chip test environment. In *The 10th IEEE European Test Symposium (ETS 2005)*, pages 2–7, May 2005.

[74] E. G. Ulrich and T. Baker. Concurrent simulator of nearly identical digital networks. In *IEEE Transactions On Computers*, volume 7, pages 39–44, 1974.

[75] A. K. Varshney, B. Vinnakota, E. Skuldt, and B. Keller. High performance parallel fault simulation. In *The IEEE International Conference on Computer Design 2001 (ICCD 2001)*, pages 308–313, 2001.

[76] I. Voyiatzis, D. Gizopoulos, and A. Paschalis. Accumulator-based test generation for robust sequential fault testing in dsp cores in near-optimal time. In *The IEEE Transactions on VLSI Systems*, volume 13, pages 1079–1086, September 2005.

[77] J. A. Waicukauski and et al. Fault simulation for structured vlsi. In *The IEEE VLSI Systems Design Conference*, pages 20–32, December 1985.

[78] S. Wang and S. Gupta. Atpg for heat dissipation minimization during scan testing. In *The IEEE/ACM Design Automation Conference (DAC 1997)*, pages 614–619, 1997.

[79] L. Wu and D. M. H. Walker. A fast algorithm for critical path tracing in vlsi digital circuits. In *The 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2005)*, pages 178–186, October 2005.

[80] G.Q. Zhang. "more than moore" - the changing international landscape, strategy and solutions of micro/nanoelectronics. In *The 8th International Conference on Electronic Packaging Technology (ICEPT 2007)*, page 1, Shanghai, August 2007.

# ACKNOWLEDGEMENTS

I WOULD LIKE TO THANK everybody who helped me during my PhD studies and without whom this work would never appear.

Particularly, I would like to express my gratitude to my supervisor prof. Peeter Ellervee for providing challenging tasks to look beyond my impossible. I have always appreciated his support and wise comments. I don't have words to express my acknowledgement to my co-supervisor prof. Raimund-Johannes Ubar for long dicussions, which helped me see the world in completely different perspective. His wise directions and strong support helped me to make this work the best way I can.

This thesis has been written in one of the best departments of Tallinn University of Technology - Department of Computer Engineering. Its motivational environment and friendly atmosphere are the key factors of success of this work. I'd like to express my special gratitude to Margus Kruus and Heljo Saar for their help and support during my PhD. studies.

I'd like to thank all the people who worked with me in the Department of Computer Engineering for their amicability and help. I'd like to express special thanks to my great colleagues Vadim Pesonen, Dr. Dmitri Mihhailov, Mairo Leier, Priit Ruberg, Dr. Maksim Jenihhin, Dr. Uljana Reinsalu, Dr. Sergei Devadze, Dr. Sergei Kostin and Dr. Marina Brik for their help and motivation to finish this work. I would also like to thank Gert Jervan, Jaan Raik and Thomas Hollstein for their helpful advises and support.

Furthermore, I would like to acknowledge the support of my PhD studies by the following organizations: Tallinn University of Technology, National Graduate School in Information and Communication Technologies (IKTDK), EU's FP7 collaborative research project DIAMOND, European Regional Development Fund through the Centre for Integrated Electronic Systems and Biomedical Engineering (CEBE), Estonian IT Foundation (EITSA) and Information Technology Foundation for Education (HITSA).

Finally I would like to thank my parents for their support and motivation. I would also like to thank my family, especially my beloved wife Tatjana for her understanding and patience. Thank you!

*Maksim Gorev*
*Tallinn, October 2015*

## ABSTRACT

This thesis addresses a series of closely related problems regarding development of BIST for high-performance pipe-lined designs. These problems can be divided into three groups covering: (1) design of benchmark circuits to represent a special class of objects to be tested (high-performance pipe-lined architectures), (2) the new methods for testing of this class of objects (BIST), and (3) the new methods for evaluating the quality of test solutions (fault simulation).

In order to find the relations between different design decisions and their corresponding testability characteristics the benchmark suite was formed of eight circuits with the same functionality, but different structures. The thesis describes the structural characteristics of the circuits and provides an overview and the discussion of their testability characteristics.

A new approach for self-testing of digital systems with pipe-lined architectures is also proposed. This is a new at-speed functional BIST methodology for these architectures. The key aspects include using inherent functionality of the system to generate test sequences and usage of MISR monitors for testing. This also leads to exploration of the potential of digitized analog signals to be used as a test-sequences for at-speed BIST. Along with that a novel evaluation environment to transfer sequential fault simulation task into a set of combinational subtasks is developed. Its goal is to speed up the process of BIST design. The methodology is evaluated in a case-study, using the benchmarks proposed previously and the results are presented and discussed.

The combinational fault simulation using parallel pattern exact critical path tracing is extended to run on multi-core systems. The parallelism is achieved in three dimensions: faults, patterns and model. The fault and pattern parallelism are achieved on each single core and the model is divided between different cores to make a fault reasoning concurrently. The experimental results using ISCAS and ITC benchmarks are presented and discussed.

Finally the novel method for observability improvement inside the sequential circuits is presented. It is shown that using only two rules to insert MISR monitors enables combinational fault simulator to be used for simulation of sequential circuits. The results of theoretical experiments to estimate performance benefits of such a method is also presented and discussed.

# ANNOTATSIOON

Käesolevas väitekirjas uuritakse probleemide kompleksi, mille eesmärgiks on isetestimise meetodite väljatöötamine ja uurimine kõrgjõudlus-konveierarhitektuuriga skeemide jaoks. Omavahel tihedalt seotud probleemide kompleks jaguneb kolme gruppi: (1) uuritavate testimisobjektide klassi defineerimine ja vastavate näidisskeemide disain (kõrgjõudlus-konveierarhitektuuriga skeemid), (2) uued meetodid nimetatud objektide klassi testimiseks (isetestimine) ja (3) uued meetodid testimiskvaliteedi hindamiseks (rikete simulatsioon).

Selleks, et leida seaduspärasusi erinevate disainiotsuste ja disainitud skeemide testitavuse vahel, genereerisime töögruppis terve seeria samasuguse funktsionaalsusega, kuid erineva struktuuriga, arhitektuure ja skeeme. Väitekirjas kirjeldatakse nimetatud skeemide strukruursete karakteristikute analüüsi ning antakse ülevaade eksperimentaalsest uurimistööst, mille tulemusena avastati rida olulisi korrelatsioone skeemide struktuursete ja testitavuskarakteristikute vahel.

On välja töötatud uus meetod konveierarhitektuuriga digitaalsüsteemide funktsionaalseks isetestimiseks. Meetodi oluliseks omaduseks on võimalus testida süsteeme reaalsel töökiirusel ja normaalsetes töötingimustes, mis tagab kõrgema testimiskvaliteedi kui traditsioonilised staatilised meetodid. Võtmeaspektiks on skeemi funktsionaalsuse ärakasutamine testjadade automaatseks genereerimiseks, kus erinevalt tuntud isetestimise meetoditest pole selleks vaja lisaaparatuuri. Ainsa riistvaralise täiendusena viiakse skeemi spetsiaalsed MISR-tüüpi registrid, milliste arvu minimeerimiseks on välja töötatud optimeerimismeetod. Esmakordselt on välja pakutud idee testida digitaalskeeme analoogsignaalide abil. Eksperimentaalne uurimistöö ja võrdlus traditsioonilise digitaalse testimisega demonstreeris uue idee suurt perspektiivsust.

Isetestimismeetodite kvaliteedi analüüsiks ja hindamiseks loodi vastav uus simuleerimiskeskkond, kus põhikriteeriumiks, tulenevalt vajadusest analüüsida väga pikki testsignaalide jadasid, oli rikete simuleerimise jõudlus. Simuleerimiskiiruse tõstmiseks teisendati mahukas järjestikskeemide simuleerimisülesanne lihtsamate kombinatoorsete ülesannete kogumiks. Uus lähenemisviis aitas kiirendada simuleerimist kaks suurusjärku.

Testide kvaliteedi analüüsi kiiruse tõstmiseks töötati välja uus meetod kombinatoorsete skeemide rikete simuleerimiseks multiprotsessorsüsteemides. Meetodi iseärasuseks on arvutusprotsesside paralleliseerimine esmakordselt kolmes dimensioonis: rikete skaala, testvektorite massiiv ja simuleeritav skeemimudel. Rikete ja vektorite analüüsi paralleelsus on saavutatud iga üksikprotsessori suhtes ning skeemimudeli analüüsiprotsess on jaotatud erinevate protsessorite vahel. Eksperementide tulemused demonstreerisid 5-10kordset simuleerimiskiiruse kasvu.

Töö viimases peatükis üldistatakse kombinatoorsete skeemide jaoks välja töötatud rikete simuleerimismeetod kasutamiseks järjestikskeemides. Selle võimaldamiseks töötati välja meetod järjestikskeemide testitavuse parandamiseks, mis seisneb kahe erireegli kasutamises vajalike täiendavate testpunktide lisamiseks signaalide monitoorimise eesmärgil. Eksperimentaalselt näidati, et väga väikese aparatuurse täienduse abil on võimalik dramaatiliselt tõsta rikete simuleerimiskiirust järjestikskeemides.

# CURRICULUM VITAE

## PERSONAL DATA

Name: Maksim Gorev
Date of Birth: July 17, 1982
Place of Birth: Narva, Estonia
Citizenship: Estonian

## CONTACT DATA

Address: 15a Akadeemia St., 12619 Tallinn, Estonia
Phone: +372 620 2265
E-mail: maksim.gorev@ttu.ee

## EDUCATION

| | |
|---|---|
| 2008 – ... | PhD studies in Information and Communication Technology, Tallinn University of Technology (TUT) |
| 2007 – 2009 | M.Sc. in Computer Science, Tallinn University of Technology |
| 2002 – 2007 | B.Sc. in Computer Science, Tallinn University of Technology |
| 1997 – 2000 | Upper Secondary Education from Narva Kreenholm Gymnasium |
| 1988 – 1997 | Secondary Education from Narva Kreenholm Gymnasium |

## CAREER

| | |
|---|---|
| 2011 – ... | Tallinn University of Technology, Faculty of Information Technology, Department of Computer Engineering; Early Stage Researcher |
| 2008 – 2011 | Tallinn University of Technology, Faculty of Information Technology, Department of Computer Engineering; Engineer |
| 2007 – 2008 | Tracking Center OÜ; Service manager |
| 2006 – 2007 | Tracking Center OÜ; Service engineer |
| 2006 – 2006 | Nixor AD AS; Service engineer |

# ELULOOKIRJELDUS

## ISIKUANDMED

Nimi: Maksim Gorev
Sünniaeg: 17. juuli, 1982
Sünnikoht: Narva, Eesti
Kodakondsus: Eesti

## KONTAKTANDMED

Aadress: Akadeemia tee 15a, 12619 Tallinn, Eesti
Tel.: +372 620 2265
E-post: maksim.gorev@ttu.ee

## HARIDUSKÄIK

| | |
|---|---|
| 2009 – ... | doktoriõpe, info- ja kommunikatsioonitehnoloogia õppekava, Tallinna Tehnikaülikool (TTÜ) |
| 2007 – 2009 | Tallinna Tehnikaülikool, arvuti ja süsteemitehnika. Magistriõpe |
| 2002 – 2007 | Tallinna Tehnikaülikool, arvuti ja süsteemitehnika. Bakalaureuseõpe |
| 1988 – 2000 | Narva Kreenholmi Gümnaasium |

## TEENISTUSKÄIK

| | |
|---|---|
| 2011 – ... | Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutitehnika instituut, Arvutisüsteemide projekteerimise õppetool; noore |
| 2008 – 2011 | Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutitehnika instituut; insener |
| 2007 – 2008 | Tracking Center OÜ; hooldusjuht |
| 2006 – 2007 | Tracking Center OÜ; hooldusinsener |
| 2006 – 2006 | Nixor AD AS; hooldusspetsialist |

# APPENDIX A

# A Benchmark Suite for Evaluating the Efficiency of Test Tools

H. Kruus, R. Ubar, P. Ellervee, M. Gorev, V. Pesonen, S. Devadze, E. Orasson, M. Brik, M. Min, P. Annus, M. Kruus, K. Meigas

*TTU, Ehitajate tee 5, 19086 Tallinn, Estonia,*
*E-mails: {helena_k, raiub, lrv, mgorev, vadim, serega, elmet, brik}@ati.ttu.ee,*
*{min, annus}@elin.ttu.ee, kruus@cc.ttu.ee, kalju@cb.ttu.ee*

**ABSTRACT: We propose a benchmark suite for systematic evaluation of efficiency of new CAD and test algorithms. The suite consists of a set of high performance signal processors. Differently from all other existing benchmark suites, all the member processors of this family perform the same function, but are implemented in different ways, differing mainly in sharing of computing resources. The circuits are characterized by different structural complexities measured in the number of reconvergent fan-outs. The latter feature has the main impact to the testability of circuits, influencing directly on the efficiency of test tools and on the quality of the given test set. The main advantage of the benchmark suite, compared to the existing ones, relies in the possibility to create systematic dependencies of the efficiency of test algorithms or test quality as a function of the structural complexity of circuits.**

## 1. Introduction

The development of CAD tools requires benchmark circuits to experiment with new algorithms and methods. Most public domain benchmarks are typically too small or simply not of the right size to give realistic assessments of the performance of new algorithms, methods and prototype tools, especially when the test problems are targeted. A number of efforts have been made to assemble public domain benchmarks [1-5], but they are not well suited for systematic evaluation of special properties of CAD and test algorithms like sensibility to different structural and architectural features of circuits like hierarchy, sharing of resources, number of reconvergent fan-outs etc.

In this paper, we propose a suite of signal processor benchmarks developed for biosignal processing in the broad field of biomedical engineering. All the circuits perform the same function but they are implemented in different ways differing mainly in sharing of computing resources. The circuits are characterized by different structural complexities measured in the number of reconvergent fan-outs. The latter feature has the main impact to the testability of circuits, to the efficiency and productivity of CAD tools like test generation or fault simulation, and to the properties of test sequences like test length or fault coverage.

The paper is organized as follows. In Section 2, a general overview about the functionality of the benchmark suite is given, and in Section 3, the structural diversities and characteristics of the circuits are presented. Section 4 provides a case study of circuits regarding different testability features, Section 5 discusses the experimental results, and Section 6 concludes the paper.

## 2. Overview of the functionality of the suite

In biomedical engineering, bioimpedance is a term used to describe the response of a living organism to an externally applied electric current. Measurement of electrical bioimpedance enables to characterize tissues and organs, to get diagnostic images, etc. [6]. Multi-channel data-acquisition devices are used often in biomedicine to measure the properties of organs and tissues. The main reason is the fact that the useful information is hidden under background signals generated by the normal body activity [7]. An example would be respiration generated noise when measuring heart activities. Electrical bioimpedance is determined by measuring of voltage response to the excitation current flow through the tissue or organ.

The impedance of tissues and organs is measured between the electrodes having different locations. Multisite and multifrequency bioimpedance information has a great diagnostic value [8 ,9].

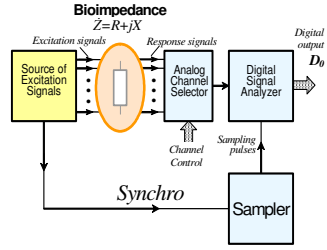



**Fig. 1.** *A simplified block diagram of DMBA*

In the following, the DSP (digital signal processing)-based solution for a multi-frequency measurement unit prototype has been described. The basic architecture of the digital multichannel bioimpedance analyzer (DMBA) is shown in *Fig.1* [7], where the bioimpedance is calculated as

$$\dot{Z}=R+jX$$

where *R* is the real part and *X* is the imaginary part.

The parameters of the response receiving part (multiplexer and signal analyzer) are defined by the use of single analog-to-digital converter (ADC) for multichannel measurements. For instance, practical measurement on body surface (thorax EBI measurements) with 8 excitation sources and also 8 response signals require operating in the frequency range of interest between 30 kHz and 100 kHz. The task can be accomplished using single ADC with at least 10MHz multisampling rate. The resolution must be between 18 to 20 to represent low (0.01% range) impedance changes adequately [10]. The heart of the electronic test arrangement (prototype) is the Field Programmable Gate Array (FPGA) Spartan™-3 from Xilinx. The FPGA handles input channel selection, sampling pulse generation, preamplifier gain control, compensating voltage code generation, reading samples from ADCs (analog-to-digital converters). The functional block diagram of the FPGA unit is shown on *Fig. 2* [10].

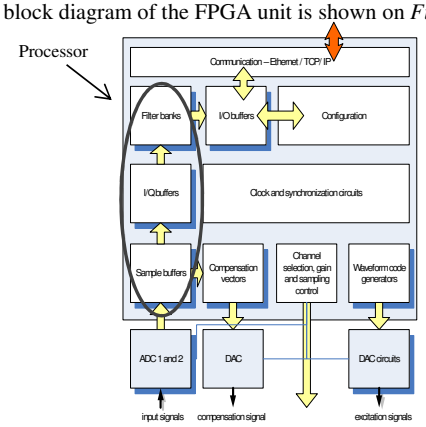



**Fig. 2.** *Functional block of the FPGA unit with I/O connections and peripheral components*

## 3. Structural characteristics of the circuits

There has been a number of different modifications developed in order to compare different architectural solutions of this functionality.

*Fig. 3* shows a design example – reconfigurable multi-channel multi-frequency application specific signal processor (DSP) that was developed for acquiring and measuring of bioimpedance signals.

The acquired data is first sampled, then sorted and finally processed. The processing includes calculating the values that are needed to calculate the bioimpedance of the tissue. The sampling order is controlled by a programmable decoder, implemented in block RAM. The sampled data is then either inserted into input buffers implemented as on-chip memory banks (see *Fig.3*) or skipped at all as in the case of the initial design. Data is accumulated and accumulation registering is performed. After that, data from registered accumulators are multiplexed to a single output register.

The general structure of the 8-channel signal processor for bioimpedance measurements is shown in *Fig. 4*.

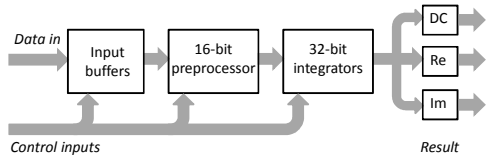



**Fig. 4.** *General structure of the bio-signal processor*

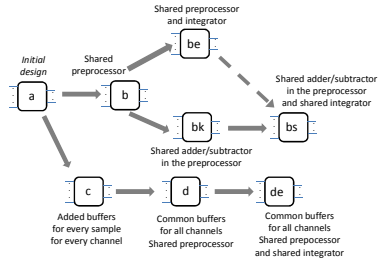



**Fig. 5.** *Overview of the benchmark designs*

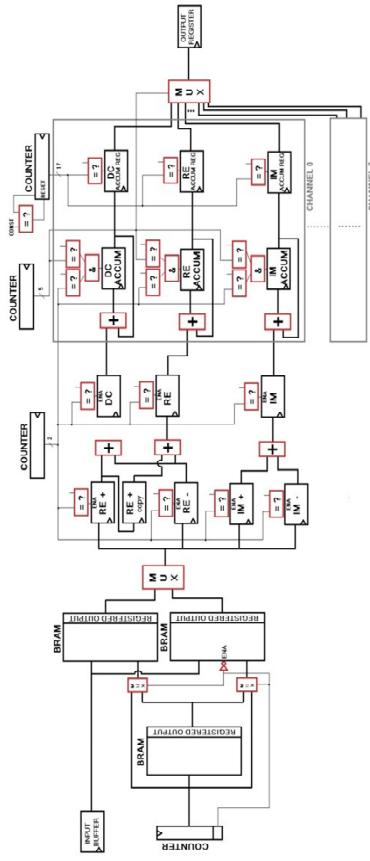



**Fig. 3.** *Structural schematic of the design*

During the design process, alterations were made to both preprocessor part of the design and the integrator part of the design, resulting in eight different configurations performing the same function: *8a, 8b, 8be, 8bk, 8bs, 8c, 8d* and *8de*. *Fig. 5* shows which successive changes were introduced into the designs. The design *8a* was the initial version with 8 data channels.

The goal of the research was to investigate how different structural implementations would impact on the testability of the design, and to find out which properties of the design will cause worse testability.

## 4. Experimental study of the benchmark suite regarding testability properties

Testability analysis of different configurations of the biosignal processing design was performed by using deterministic and pseudorandom test pattern generators [11], fault simulator [12] and by using the algorithms for hybrid BIST optimization developed in [13]. Several testability characteristics presented in *Table 1* were analyzed: the deterministic test length achieved (DTL) and the needed time for deterministic test pattern generation (DTG), the time needed for fault simulation (FS) and for the pseudorandom test generation (RTG), the hybrid BIST length (HBL) and the calculated optimal test cost of hybrid BIST (HBC). Generation times are given in seconds, test lengths in numbers of patterns, and costs in abstract units. The changes in testability characteristics for the benchmark suite are shown in *Fig. 6.*

**Table 1.** *Testability characteristics of signal processors*

| Design | DTL | DTG | FS | RTG | HBL | HBC |
|---|---|---|---|---|---|---|
| 8a | 1364 | 47 | 13.7 | 1408 | 23 038 | 197 823 |
| 8b | 1201 | 34 | 11.8 | 1130 | 18 540 | 138 324 |
| 8bk | 1288 | 35 | 11.3 | 1129 | 17 497 | 144 876 |
| 8c | 1320 | 75 | 15.5 | 1583 | 35 641 | 224 121 |
| 8d | 1394 | 62 | 16.6 | 1647 | 32 610 | 209 384 |
| 8be | 995 | 114 | 27.9 | 2784 | 14 202 | 104 474 |
| 8de | 1096 | 112 | 33.4 | 3344 | 33 968 | 162 557 |
| 8bs | 1186 | 296 | 69.0 | 7095 | 14 086 | 113 038 |

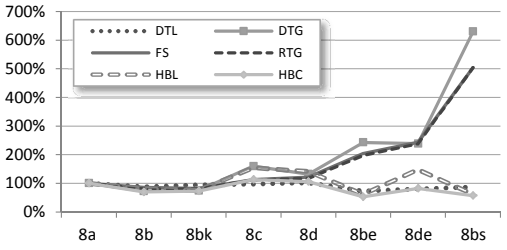



**Fig. 6.** *Changes in testability characteristics*

The different design implementations are characterized by different levels of sharing of resources such as input buffers, preprocessing units, adders and subtractors in preprocessors, and integrators. Sharing of the resources was accompanied by introducing additional multiplexers and control circuits which in their turn increased the number of reconvergent fan-out branches in the topology of the circuits. A rough estimation of the number of convergent control signals is given in *Table 2*.

**Table 2.** *Modifications in the different benchmark designs*

| Design | # of reconvergent control signals | | | Modifications made |
|---|---|---|---|---|
| | Preprocessor | Integrator | Total | |
| 8a | 32 | 64 | 96 | Initial design |
| 8b | 32 | 64 | 96 | Shared preprocessor |
| 8bk | 64 | 64 | 128 | Shared preprocessor and adder/subtractor |
| 8c | 32 | 64 | 96 | Initial design with input buffers |
| 8d | 32 | 64 | 96 | Shared preprocessor |
| 8be | 32 | 512 | 544 | Shared preprocessor and integrator |
| 8de | 32 | 512 | 544 | Shared preprocessor and integrator |
| 8bs | 64 | 1536 | 1600 | Maximum sharing of resources |

## 5. Discussion of the results

The changes in design alternatives are characterized by different structural complexities which will have a direct impact on testability of circuits and on the testing quality. The experimental results presented in *Table 1* and *Fig. 6* allow easily to create functional dependencies between the testability features and the resource sharing options in design alternatives, which allows to find proper tradeoffs. In the following we discuss in details the depicted results.

The transition from *8a* to *8b* laid in replacement of 8 channels in preprocessing part of the circuit by a single common channel, thus removing the redundancy by a shared preprocessor. This resulted in improvement of all the testability characteristics. The best improvements were in reduction of test synthesis time (for deterministic test 1.4 and for pseudorandom test 1.25 times). Fault simulation became 1.16 times faster. The cost of the hybrid BIST significantly improved – one of the reasons is smaller number of inputs in *8b*, which results in the less cost of the memory component of the BIST.

The transition from *8b* to *8be* was equivalent to the replacement of 8 channels of integrators with a single channel. Multiplexers were added to the inputs of adders in the integrator. The deterministic test length improved - it was 1.2 times shorter, which can be explained by the reduction of the circuit complexity. On the other hand, the time needed for deterministic test generation was 3.35 times higher because of the increased number of reconverging control signals in the circuit from 94 to 544, which causes higher number of backtracks during search for consistent solutions. Also, fault simulation time became 2.36 times slower, and the time needed for pseudorandom test generation was 2.46 times higher. This is explained by the use of exact critical path tracing algorithm [12] used for fault simulation which is highly

sensitive to the number of reconvergent control signals. Since pseudorandom test generation uses the same fault simulator, the test generation time consequently as well increases. The cost of the hybrid BIST was improved due to the smaller number of deterministic vectors needed.

The transition from *8b* to *8bk* consists in using one adder/subtractor and additional multiplexers in the preprocessor part. The increase of the reconvergent control signals (from 96 to 128) did not significantly influence the testability of the circuit.

The transition from *8bk* to *8bs* combined the preprocessor part of the design *8bk* and the integrator part of the design *8be*. The number of reconvergent control signals increased drastically (128 for *8bk* and 1600 for *8bs*). *Fig. 6* shows worsening of the testability regarding test generation and fault simulation: the time of deterministic test generation became 8.45 times longer, the time of fault simulation 6.1 times longer, and the time of random test generation became 6.28 times longer. On the other hand, because of the reduction in circuit size, the length of deterministic test set became slightly shorter (1.08 times). The length of optimal hybrid BIST was 1.24 times shorter and optimal cost 1.28 times smaller due to the smaller number of seeds for LFSR.

The transition from *8a* to *8c* resulted in implementing the programmable input buffers. The 8 channels of data remained. In *Fig. 6* it can be seen that the time related characteristics have become worse: the generation time for deterministic test became 1.59 times longer, and for random test 1.13 times longer. The fault simulation became 1.13 times slower. This worsening of indicators can be explained by the increase of the number of reconvergencies because of adding control signals for addressing the buffer registers. The test length did not change because of the circuit size remained the same. The length of hybrid BIST sequence test became 1.54 times longer, and the cost of Hybrid BIST was bigger for *8c* due to the bigger number of inputs (buffer registers).

In the transition from *8c* to *8d* eight channels of the preprocessor were removed and replaced with a single channel. Multiplexers were added to the inputs of the preprocessor. The characteristics that changed most significantly were deterministic test generation time (became shorter) and the length of the optimal hybrid BIST became slightly shorter, similarly as in the case of "from *8a* to *8b*".

In transition from *8d* to *8de*, 8 channels of integrator were replaced by a single channel, and few multiplexers were added, which caused the increase of reconvergent control signals (*Fig.6*), and longer times for test generation and fault simulation: for deterministic test 1,81 times and for random test 2.03 times longer. Fault simulation became 2.01 times slower. The deterministic test set was 1.27 times shorter and the cost of the BIST reduced 1.28 times (due to the smaller number of seeds). This case affected the testability characteristics in the similar way as in the case from "*8b* to *8bk*".

## 6. Conclusions

As the result of the cooperation in the fields of computer, electronics and biomedical engineering in the Estonian Research Excellence Centre CEBE, a benchmark suite was developed for evaluating the CAD tools in their efficiency and quality in designing dependable digital systems.

Differently from all other existing benchmark suites, all the member processors of this family perform the same function, but are implemented in different ways, differing mainly in sharing of computing resources. This gives an excellent possibility for direct systematic characterization of CAD tools by creating functional dependences for different testability markers on the structural complexity of circuits. Existing benchmark suites do not provide such a possibility.

By experimental research, a correlation was established between the structural properties of circuits and their testability characteristics. It was shown that sharing of resources in designs, which leads to increasing numbers of fan-out reconvergencies, may reduce the test length, but on the other hand, will increase the time of test synthesis, and may reduce the test quality.

A useful synergy was achieved by creating a selection of biosignal processors, which will have practical use in medical field, but which simultaneously can be used as well as a family of benchmark circuits for analyzing the properties of new test algorithms in the field of electronics.

Acknowledgement: The work has been supported by the project FP7-ICT-2009-4-248613 DIAMOND and by EU through the European Regional Development Fund.

### References

[1] F. Brglez, H. Fujiwara. A Neutral Netlist of 10 Combinational Benchmark Circuits. ITC, 1985, pp.785-794.

[2] F. Brglez, D. Bryan, K. Kominski. Combinational Profiles of Sequential Benchmark Circuits. ISCAS, 1989, pp.1929-1934.

[3] MCNC Benchmarks Suite. http://sites.google.com.site/xiaoleicustc /research/mcnc

[4] HLSynth92 benchmarks family. www.cbl.ncsu.edu:16080 /benchmarks/HLSynth92

[5] ITC'99 Benchmarks webpage, CAD Group, Politecnico Torino, www. cad.polito.it/tools /itc99.html

[6] E.T.McAdams, J.Jossinet. Tissue impedance: a historical review. Physiological Measurements, Vol. 16, pp. A1-A13.

[7] V. Pesonen, M. Gorev, P. Annus, M. Min, P. Ellervee. Reprogrammable Data Acquisition Unit to Reduce Aliasing Effect in Bioimpedance Measurements. 7th Annual FPGAworld Conf., Copenhagen, Denmark, 6 pp., Sept. 2010.

[8] S. Grimnes, O. G. Martinsen. Bioimpedance and Bioelectricity Basics. Academic Press, San Diego, 2000.

[9] T. Dudykevych, et al. Impedance Analyser Module for EIT and Spectroscopy Using Undersampling. Physiological Measurement, No. 22, Institute of Physics Publ. Ltd, UK, pp. 19-24, 2001.

[10] M.Min, T.Parve, P.Annus, T.Paavle. "A method of synchronous sampling in Multifrequency Impendace Maesurments". IMTC 2006, Sorrento, Italy, Apr. 2006.

[11] J. Raik et al. Turbo Tester Manual. Tallinn University of Technology, www.pld.ttu.ee/TT.

[12] R.Ubar, S.Devadze, J.Raik, A.Jutman. Parallel X-Fault Simulation with Critical Path Tracing Technique. DATE-2010, Dresden, Germany, March 8-12, 2010, pp. 1-6.
[13] H.Kruus. Optimization of BIST in Digital Systems. PhD Thesis. Tallinn University of Technology, 2011, 161 p.

# APPENDIX B

# At-Speed Self-Testing of High-Performance Pipe-Lined Processing Architectures

Maksim Gorev, Raimund Ubar, Peeter Ellervee, Sergei Devadze, Jaan Raik, Mart Min

*Tallinn University of Technology, Akadeemia tee 15A, 12618 Tallinn, Estonia,*
*E-mail: {maksim.gorev, raimund.ubar, peeter.ellervee, sergei.devadze, jaan.raik}@ati.ttu.ee, min@elin.ttu.ee*

**ABSTRACT:** We propose a new methodology for Built-In Self-Test (BIST) where contrary to the traditional scan-path based logic BIST, the proposed solution for test generation does not need any additional hardware, and will not have any impact on the working performance of the system. A class of digital systems organized as pipe-lined signal processing architectures is targeted. The data used for processing in the system are used as test pattern sources. Testing at normal working conditions, and with typically processed data, allows exercise the system on-line and at-speed, facilitating the detection of dynamic faults like delays and cross-talks to achieve high test quality. The proposed new self-test method is free from the negative aspect of over-testing, compared to the traditional logic BIST approaches, and uses a minimal amount of additional hardware. Experimental research was based on the case study of a specialized bio-signal processor architecture, and the results showed promising results in reducing the cost of testing and achieving high fault coverage.

**Keywords:** pipe-lined signal processing architectures, built-in self-test, at-speed testing, design for testability

## 1 Introduction

The technology advancements impose new challenges to testing modern chips as device geometries shrink, and deep-submicron delay defects are becoming more and more important requiring more accurate dynamic tests than before [1]. Therefore testing of chips in dynamics by so called at-speed test is becoming the must.

Increasing size and complexity of digital systems directly reflects in more demanding test generation and application strategies. The use of scan chains has proven to be often inadequate increasing the cost in terms of additional hardware and testing time [2], excessive power dissipation during test [3] and leading to yield loss because of over-testing [4].

A lot of research has been carried out to relieve the burden of external testers by introducing system self-test approaches like hardware-based logic Built-in Self-Test (LBIST) which typically use Linear Feedback Shift Registers (LFSR) [5]. In LBIST the typical functions of external test equipments like test generation and response analysis are carried out on-chip, so that the tester should not handle high-speed signals externally and its role should remain only to send the test enable signals to the chip under test, and to receive the pass/fail signals. For example, scan-based and logic BIST solutions such as [6] relax the requirements on testers and considerably reduce the overall testing cost.

An important trend today is the at-speed test [7] having additional benefit of the ability to test circuits under conditions that are as close as possible to normal circuit operation. This factor has a direct impact on the number of chips that are found defective during system operation but still pass all manufacturing and functional tests. At-speed testing can be used for characterization and can also expedite test application time.

The question is whether a self-test sequence running in the system can adequately exercise its hardware components satisfying the targeted fault coverage requirements. Achieving the test quality target requires application of proper test sequences which is the focus of the current paper. It should also be pointed out that the quality of a test is measured not only by its fault coverage, but also by its code size (to be stored in the memory of the chip), hardware overhead, and by the test execution time.

The goal of the paper is to propose an approach which combines the ideas of traditional LBIST with at-speed testing to improve the test quality at less testing overhead and avoiding performance loss compared to the traditional self-test approaches. The feasibility and efficiency of the new method is demonstrated for a particular class of pipe-lined processing architectures which are easily adaptable for at-speed on-line self-testing by inherent functional sequences.

The rest of the paper is organized as follows. In Section 2, an overview about the state-of-the-art is given, Section 3 presents the general idea and the scheme of the proposed method, followed in Section 4 by the description of the representative case study design. Section 5 presents the results of experimental research, and Section 6 concludes the paper.

## 2 State-of-the-art of self-test techniques

In traditional LBIST, test pattern generation is mostly performed by Linear Feedback Shift Registers (LFSR) [5], cellular automata [8] or multifunctional registers like BILBO (Built-in Logic Block Observer) [5] to apply pseudorandom patterns to the Circuit Under Test (CUT) and to analyse its output responses. However, many circuits contain random-pattern-

resistant faults, which limit the fault coverage that can be achieved with this approach.

One method to improve the fault coverage for LBIST is to modify the CUT by either inserting test points [4] or by redesigning it to improve the fault coverage [6]. The drawback of these techniques is that they generally add additional logic levels to the circuitry that can degrade system performance. Another possibility to improve the fault coverage is to use weighted pseudorandom test sequences [9]. The disadvantage of this approach is in the need of storing of the weight sets on chip, and also, a dedicated control logic is required to switch between weights, so the hardware overhead may become large.

A "mixed mode" approach, where deterministic patterns will be added to detect hard-to-test faults, has been developed in [10-13]. In [10] a technique based on reseeding LFSR was proposed that reduces the storage requirements. In [11], multi-polynomial LFSR for encoding a set of deterministic test cubes was introduced, and in [12] a technique called bit flipping for generating deterministic test cubes using BIST control logic was proposed. Further, in [13] a mixed-mode approach was presented in which deterministic test cubes are embedded in the pseudorandom sequence of bits itself.

Established BIST solutions use special hardware (typically LFSR) for pattern generation (TPG) and test response evaluation (TRE) on chip [5], but this in general introduces significant area overhead and performance degradation. To overcome these problems, specialized methods were proposed which exploit specific functional units such as arithmetic units for on-chip test pattern generation [14,16], which may afford to reach similar fault coverage like traditional LFSR-s. These methods are called arithmetic BIST (ABIST), since they essentially adopt the additive congruential generation scheme of pseudo-random numbers [17].

In [18,19], a mixed-mode or hybrid BIST approach was proposed, where a test set is assembled from two parts, from pseudorandom test patterns that are generated on-line, and deterministic test patterns that are generated off-line and stored in the system. Combination of both test sources in an optimized fashion allowed to improve the traditional LBIST in targeting hard-to-test faults. A similar approach called hybrid functional BIST (HyFBIST), where instead of LBIST the inherent functional sequences were used, was proposed in [20,21] for testing digital systems, and particularly micro-programmed data-paths.

In this paper we generalize and combine the ideas of using inherent functional blocks for test generation [14,15] and the inherent working sequences produced by the UUT itself for self-testing purposes. We propose an overall functional self-test concept for pipelined architectures where the working sequences are produced on the primary inputs of the system, and the internal signals in selected test-points are monitored by Multiple Input Signature Analysers (MISR). We propose a systematic procedure for selecting the test-points to achieve the best overall fault coverage at minimum testing overhead and cost.

## 3 General description of the method

Consider a digital system as a network of sub-circuits (blocks) where all the blocks may play simultaneously two roles: on one hand, each block will be itself the unit under test (UUT), and on the other hand, it will serve as the test pattern generator for the subsequent blocks it is feeding. As the overall test source, selected input working sequences (as functional test) will be used.

Two main problems arise: (1) how to find the best functional test sequences, and (2) how to find the minimal set of test-points for monitoring to achieve the highest fault coverage of testing.

In some cases, the first problem can be solved straightforwardly like in the instruction set architectures or in signal processing units. In the first case, the instructions can be exercised one by one where the problem recedes to finding only proper data (operands) as test patterns [22-23]. In case of signal processing units, the proper signal types to be processed in the working modes, and used as well for testing purposes, may be selected case by case by trial and error methods.

The second problem of selecting test-points for monitoring depends how well are the different blocks tested by the given functional test sequences. In Fig.1, an example of a pipe-lined signal processing unit is given which is partitioned into 6 blocks. Two solutions are demonstrated for monitoring the behaviour of the circuit with MISRs. The upper solution shows the case where all blocks are monitored whereas the lower solution uses only three MISR: the first is monitoring the behaviour of B1 and B2 as the whole, the second MISR is monitoring B3, and the third one is monitoring the blocks B4, B5 and B6 as the whole.

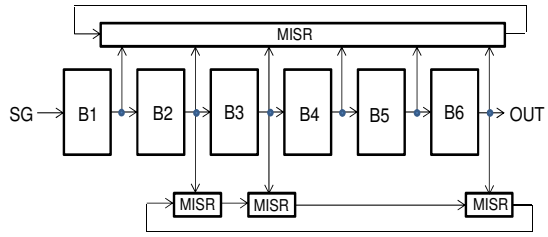



**Fig.1**. *Pipe-lined signal processing unit*

The task of partitioning of the whole system into UUT blocks has the goal to find the highest fault coverage

at the given functional test by achieving well-balanced testability at the minimum number of MISR.

To find the minimum hardware overhead we propose the following algorithm for selecting test-points.

**Algorithm 1**

1. Put MISR on the primary output of the circuit, and find fault coverage (FC) for the given test sequence.
2. If FC is sufficient, then the problem is solved.
3. Partition the circuit into a set of *n* blocks (each with its own MISR). Find FC for each block as a UUT.
4. Integrate the connected blocks with high FCs into UUTs (with a single joint MISR) to minimize the number of MISR, so that the total FC of the system remains sufficient high.
5. For the blocks with lower FC, repeat the steps 3,4 inside the block by partitioning it into smaller sub-circuits to improve the overall FC of this block.

In case when the fault coverage will not satisfy either globally for the whole circuit or for particular blocks as UUTs, either the better functional test sequences should be found, or different methods, similar to the ones for improving LBIST described in Section 2, may be used.

The proposed method facilitates the idea of LBIST strategy except using instead of dedicated test generators of LBIST the inherent test functionality of existing hardware in the system. The method affords at-speed testing with no performance degradation and with little hardware overhead and reduced test cost.

The clock cycle based observation technique allows to avoid fault masking, and to achieve high fault coverage. The test response observation is carried out using built-in MISR as the only hardware overhead.

The proposed method has several advantages compared to the state-of-the-art scan-path based LBIST methods:

(1) no hardware test pattern generators, and no scan-path for shifting in external test patterns are needed, which results in smaller overhead;
(2) compared to LBIST over-testing is avoided, since only functional working test patterns are used;
(3) testing is carried out in the normal working clock-rate which guarantees at-speed exercising the circuit.

To investigate the feasibility of the method in the sense of achieving sufficient fault coverage in real cases, we carried out experimental test research with a digital signal processor unit developed for industrial purposes for measuring electrical bio-impedance [24,25].

## 4 Signal processing unit as UUT

The unit under test (UUT) is a bio-impedance signal analyser, which implements a simplified signal processing algorithm [24]. A typical digital solution is that the response voltage is digitized in an analog-to-digital converter (ADC) into a uniformly sampled train of digital data, which is then processed numerically in a digital signal processing (DSP) unit, often using the Discrete Fourier Transform (DFT). Because the whole signal path from the generation of the set of excitation signals to the A/D conversion procedure and data analysis is synchronous by design, optimized signal processing methods can be applied. Using of sampling, which is synchronous to the known excitation waveform, enables to use a simplified but much faster signal processing than the Fourier Transformation is. When sampling the response signal uniformly with intervals $t = T/4$, where $T$ is a period of the signal, the following simple mathematics is valid [24]:

(1) the direct current component can be determined as

$$DC = (Re^{+} + Re^{-})/2 \quad \text{or} \quad DC = (Im^{+} + Im^{-})/2$$

(2) the real *Re* and imaginary *Im* parts of the phasor *Z* of complex impedance is determined as

$$Re = (Re^{+} - Re^{-})/2, \quad \text{and} \quad Im = (Im^{+} - Im^{-})/2$$

The mentioned signal analyser is a part of the developed digital multichannel bio-impedance analyser (DMBA) , which is depicted in Fig. 2.

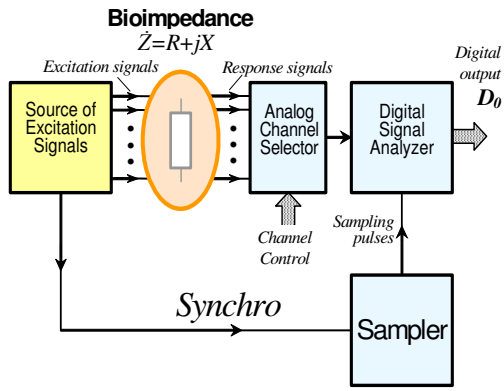



**Fig. 2.** *A simplified block diagram of DMBA*

For the test purposes of the circuit, the sampler is implemented as 80MHz clock signal and excitation signal generator along with body and analog part was exchanged with signal generator of particular type. The architecture of the signal analyser circuit is depicted on Fig. 3. As it can be seen, the circuit has a pipe-line structure in order to be implemented in a low-cost FPGA. The signal is first sampled into the input buffer. On the next stage the signals are distributed to the particular registers of the 8 different

channels. The sampling is performed on channed-after-channel basis. Every sample out of 4 samples taken per channel is saved into its corresponding 16-bit register. On the next stage Real and Imaginary and Direct current components are computed with adders and subtractors, using equations (1) and (2). On the next pipeline stage the computed components are integrated using adders and saved into 32-bit registers, called output buffers. Integration is made over a 1 millisecond period. After that the values of the output buffers are transferred to the output register of the anayser.

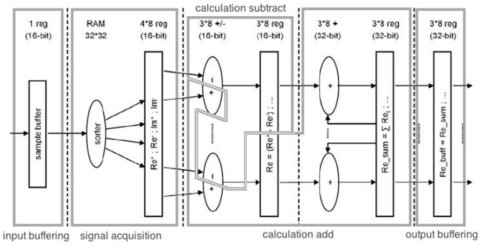



**Fig. 3.** *The architecture of Signal Analyser*

We investigated four types of signal generators for using as test sources to provide input signals to the channels of the analyser: *sine*, *chirp*, *saw-tooth*, LFSR. All the channels get the same signal, so that we can test each channel equally to each other. All the generators are implemented for the simulation environment in VHDL.

(1) The *sine* signal generator is using floating point arithmetics and *sin*() function of the VHDL math library. It can take amplitude, phase and frequency as a parameters to produce the corresponding *sine* wave. During the experiments the amplitude was set to 15 bit, taking into account 1 sign bit and 16-bit wide input of the analyser. The phase was set to 90 degrees in order to produce the input signal from the upper part of the wave. This was done because the signal would produce more unique values in less time, because it covers all the values from top to the bottom in half-period. It was useful to check whether the test sequences of small length could produce meaningful results. The frequency was modified during the experiment in order to detect the better signal for testing this device.

(2) The *chirp* generator takes as parameters start and stop frequency periods as well as number of samples in which frequency should change from start to stop frequency. The *chirp* generator changes the frequency every sample it produces. The amplitude remained 15bit + 1 sign bit and phase remained 90 degrees. During the experiments we have changed the length of the *chirp* signal – number of samples from start to end frequency.

(3) *Saw-tooth* signal is implemented as a counter. The parameter it takes is a period of the signal. The generator produces equally spaced samples of the *saw-tooth* signal of this period. The amplitude is 15bit+1 sign bit.

(4) LFSR signal generator is implemented as 16-bit linear feedback shift register. The seed is taken so that it goes through all the 65535 possible values except 0. The size of the LFSR was chosen in accordance to the input width of the signal analyser under test.

## 5 Experimental results

Experiments were carried out for Signal Analyser in Fig.3, presented as equivalent circuit with high-lighted pipe-lined tracks in Fig.4. As the result of the experimental research according to Algorithm 1, the circuit was finally partitioned into 7 blocks as separate UUTs which are characterized in Table 1.

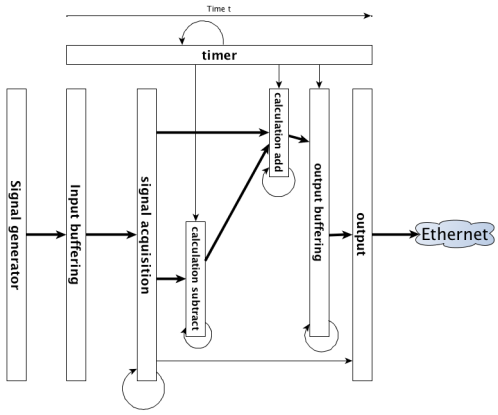



**Fig. 4.** *Equivalent circuit for the Signal Analyser in Fig.3.*

| No | Name of the block | Number of faults | Number of inputs | Number of outputs |
|---|---|---|---|---|
| 1 | *calc_add* | 69544 | 1431 | 896 |
| 2 | *calc_sub* | 18588 | 791 | 256 |
| 3 | *in_buf* | 98 | 17 | 16 |
| 4 | *out_buf* | 14750 | 1554 | 769 |
| 5 | *out* | 7480 | 709 | 64 |
| 6 | *sig_acq* | 8560 | 538 | 520 |
| 7 | *timer* | 512 | 18 | 17 |
| Total | | 119532 | 5058 | 2538 |

***Table 1.*** *Characteristics of the blocks in Fig.4*

We calculated the fault coverage for all the 7 blocks as well as the total fault coverage for four different types of signals: *sine, chirp, saw-tooth* and LFSR. The results of the experimental research in percentage of fault coverage for all the different blocks are presented in Table 2 and as the bar diagram in Fig. 5. As we see, the best results were achieved for the input

signal *sine* where the fault coverage was 98.20%. The lowest total fault coverage 75.99% was registered for the signal type *saw-tooth*.

| No | Name of the block | Input signal types | | | |
|---|---|---|---|---|---|
| | | *sine* | *chirp* | *saw-tooth* | LFSR |
| 1 | *calc_add* | 97.37 | 94.86 | 76.80 | 95.71 |
| 2 | *calc_sub* | 98.85 | 99.20 | 64.90 | 99.20 |
| 3 | *in_buf* | 82.65 | 82.65 | 82.65 | 82.65 |
| 4 | *out_buf* | 99.88 | 99.86 | 74.74 | 99.86 |
| 5 | *out* | 99.14 | 99.06 | 78.66 | 99.14 |
| 6 | *sig_acq* | 95.63 | 95.63 | 95.63 | 95.63 |
| 7 | *timer* | 94.14 | 94.14 | 94.14 | 94.14 |
| Total | | 98.20 | 96.68 | 75.99 | 97.21 |

***Table 2**. Results of fault coverage experiments*

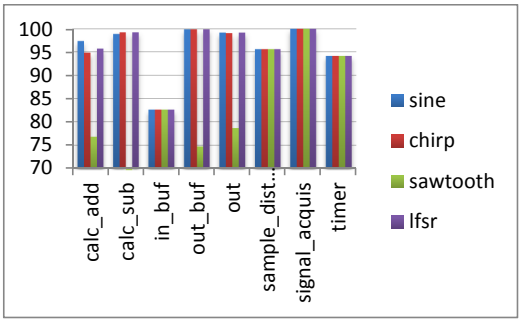



***Fig 5**. Distribution of fault coverage in the circuit*

Considering the distribution of fault coverage among different blocks we see that the lowest test quality is mapped to the block *in_buf*. Hence, for this block the improvement of the testability by any of the methods referenced above in Sections 2 and 3 can be foreseen (this task was not the goal of this case study paper). However, since the block *in_buf* is rather small (characterized by only 98 faults), the improvement of its testability will not lead to considerable increase in the total fault coverage of the whole circuit.

Since the cost of testing depends on the time used for carrying out the self-test procedure, we investigated how the fault coverage will depend on the test length measured in the number of test patterns. The results are shown as the graphics for the different four signal types in Fig. 6.

The most cost effective would be the LFSR based self-test sequence where the fault coverage around 90% will be achieved already after 80 000 test patterns (clock signals) whereas the *sine* signal based and *chirp* signal based tests achieve only about 85% and 80% fault coverage, respectively, at the same test length. When doubling, however, the test length, the *sine* based and LFSR based tests become equal at the 95% fault coverage. Especially sensitive to the length of the test is the *chirp* signal based test sequence.

We compared the test quality achieved by the proposed method with traditional scan-path (SP) techniques both for using LFSR pseudorandom and deterministic test sequences. The results are presented in Table 3.

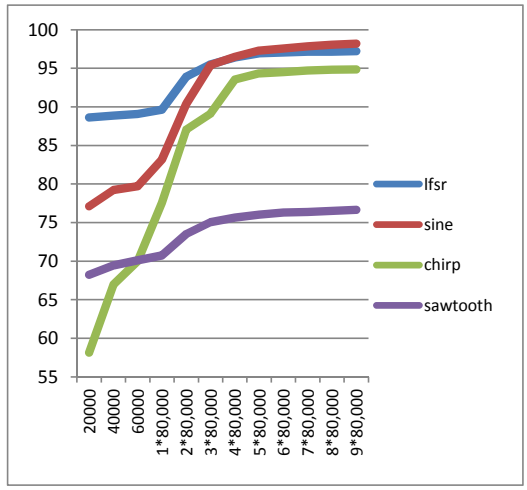



***Fig 6**. Dependence of the fault coverage on test length*

| Method | Fault coverage | Test length(TL) | Testing time (clocks) |
|---|---|---|---|
| Proposed | 97.78 | 500000 | 500000 |
| SP & LFSR | 96.82 | 500000 | x 2528 |
| Proposed | 98.27 | 1000000 | 1000000 |
| SP & LFSR | 98.73 | 1000000 | x 2528 |
| SP & deterministic | 98.69 | 1364 | x 2528 |

***Table 3**. Comparison of different methods*

As we see from Table 3, the fault coverage is nearly the same for all the methods compared. However, to get the same fault coverage as with the proposed method, the test length of the scan path & LFSR based approach should be even twice bigger compared to the proposed method. To calculate the testing time cost in clock numbers, the test length for both referenced scan-path based methods should be multiplied by the length of the scan path which is equal to 2528 bit (the total number of inputs of all the tested blocks in the given circuit). For the proposed method, the testing time in number of clocks is equal to the test length. Hence, we can conclude that the time cost of the proposed method is about 10 times cheaper than the SP & deterministic approach and more than 2500 times cheaper than SP & LFSR at the same fault

coverage (in the latter case the single scan-path was assumed).

## 6 Conclusions

We introduced a new approach to self-testing of digital systems with pipe-lined architectures using inherent functionalities of systems with capability to produce internal self-test sequences. The added value of using inherent functional self-test sequences is the higher test quality explained by on-line at-speed testing. The approach does not need to store high volume test data in the system memory. Additional hardware is as well not needed for on-line test pattern generation as in the case of traditional LBIST. The only needed additional test hardware is related to using MISR for monitoring the test responses. To minimize the needed additional MISR hardware overhead, an original algorithm for selecting test-points was developed. As the result of avoiding artificial embedded test pattern generators like in case of LBIST, and of using only normal working sequences for test purposes, the danger of over-testing and the related yield loss are removed.

To investigate the feasibility of the method to achieve high fault coverage, we carried out experimental research with a digital Signal Analyser unit as a case study, which was developed for industrial purposes for measuring electrical bio-impedance.

The goals of the experiments were twofold: (1) to select the best type of input signal for testing purposes from a set of signals typically used for processing in the given Signal Analyser, and (2) to compare the new method with traditional scan path based testing methods.

Experimental research showed that the best testing capability has the *sine* signal (with fault coverage of 98,2%) compared to the LFSR based pseudorandom (97,2%) and *chirp* (96,7%) signals at the same test length. The worse testing capability has the *saw-tooth* type signal (76%). The fault coverage achieved by the *sine* signal was 98,2%, which is nearly the same compared to the traditional scan-path pseudorandom (98,7%) and deterministic (98,7%) test approaches. The gain in testing time cost was 10 times compared to the deterministic and more than 2500 times compared to the pseudorandom single scan-path based approach.

Acknowledgements: The work was supported by the Research Excellence Centre CEBE funded by EU Structural Funds. We thank also Elmet Orasson from Tallinn University of Technology who helped us to carry out a part of the experimental research.

## References

[1] T.Mak, S.Krstic, K.-T.Cheng, L.-C.Wang. New challenges in delay testing of nanometer, multi-gigahertz designs. IEEE Design & Test of Computers, 21(3), 2004, 241-248.

[2] L.Bushard, N.Chelstrom, S.Ferguson, B.Keller. DFT of the Cell Processor and its Impact on EDA Test Software. In IEEE Asian Test Symposium, 2006, pp. 369-374.

[3] S.Wang, S.Gupta. ATPG for heat dissipation minimization during scan testing. In ACM IEEE Design Automation Conference, 1997, pp. 614-619.

[4] L.Chen, S.Ravi, A.Raghunathan, S.Dey. A Scalable Software-Based Self-Test Methodology for Programmable Processors. In IEEE/ACM Design Automation Confernce, 2003, pp. 548-553.

[5] L.-T.Wang, C.-W.Wu, X.Wen. VLSI test principles and architectures. Morgan Kaufmann, 2006.

[6] G.Hetherington, T.Fryars, N.Tamarapalli, M. Kassab, A.Hassan, J.Rajski.. Logic BIST for large industrial designs. Proc. IEEE Int. Test Conf., pp. 358–367, 1999.

[7] B.Nadeau-Dostie. Design For At-Speed Test, Diagnosis and Measurement. Kluwer Academic Publishers, 2002.

[8] P.D. Hortensius, R.D. McLeod, B.W. Podaima. Cellular automata circuits for BIST. IBM J of Research and Development. Vol.34, No.2.3, pp.389-405, 1990.

[9] M.F. AlShaibi, Ch.Kime. MFBIST: A BIST method for random pattern resistant circuits. Proc. ITC, Oct. 1996, pp.176-185.

[10] B.Koenemann. LFSR-coded test patterns for scan designs. Proc. European Test Conf., Mar. 1991, pp.237-242.

[11] S.Hellebrand, J.Rajski, S.Tarnick, B.Courtois, S.Venkataraman. Built-in test for circuits with scan based on reseeding of ‚ulti-polynomial linear feedback shift registers. IEEE Trans. On Comput. Vol. 44, pp.223-233, Feb. 1995.

[12] H.-J. Wunderlich, G.Kiefer. Bit flipping BIST. Proc. ICCAD, Nov. 1996, pp.337-343.

[13] N.A. Touba, E.J.McCluskey. Bit-fixing in pseudorandom sequences for scan BIST. IEEE Trans. on CAD of IC and Systems, Vol.20, No.4, Apr.2001.

[14] R.Dorsch, H-J.Wunderlich. Accumulator based deterministic BIST. ITC, 1998, Washington D.C.

[15] J.Rajski, J.Tyszer. Arithmetic BIST in embedded systems, Prentice-Hall, N J (1998).

[16] I.Voyiatzis, D.Gizopoulos, A.Paschalis. Accumulator-based test generation for robust sequential fault testing in DSP cores in near-optimal time. IEEE Trans. on VLSI Systems, Vol.13, No.9, Sept., 2005, pp.1079-1086.

[17] D.E.Knuth. The art of computer programming. Vol.2, Addison-Wesley, 1981.

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

[19] G.Jervan, P.Eles, Z.Peng, R.Ubar, M.Jenihhin. Hybrid BIST Time Minimization for Core-Based Systems with STUMPS Architecture. 18th Int. Symposium on Defect and Fault Tolerance in VLSI Systems. Cambridge, MA, USA, November 3-5, 2003.

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

[21] N.Mazurova, J.Smahtina, R.Ubar. Hybrid Functional BIST for Digital Systems. Proc. of the 9th Biennial Baltic Electronics Conference, Oct. 3-6, 2004, Tallinn, pp.205-208.

[22] D.Gizopoulos et al. Systematic software-based self-test for pipelined processors. IEEE Trans. on VLSI Systems, Vol. 16, No.11, Nov. 2008, pp.1441-1453.

[23] R.Ubar, V. Indus, O. Kalmend, T.Evartson. Functional Built-In Self-Test for Processor Cores in SoC. The 30th IEEE NORCHIP Conference, Copenhagen, Denmark, Nov. 12-14, 2012.

[24] P.Ellervee, P.Annus, M.Min. High Speed Data Preprocessing for Bioimpedance Measurements: Architectural Exploration. NORCHIP, 2009.

[25] H.Kruus, R.Ubar, P.Ellervee, M.Brik, M.Gorev, M.Kruus, E.Orasson, V.Pesonen, P. Annus, M.Min, K.Meigas. A Benchmark Suite for Evaluating the Efficiency of Test Tools. Proc. of Baltic Electronics Conference, Tallinn, October 3-5, 2012.

# APPENDIX C

# Functional Self-Test of High-Performance Pipe-Lined Signal Processing Architectures

Maksim Gorev, Raimund Ubar, Peeter Ellervee, Sergei Devadze, Jaan Raik, Mart Min

*Tallinn University of Technology, Akadeemia tee 15A, 12618 Tallinn, Estonia,*
*E-mail: {maksim.gorev, raimund.ubar, peeter.ellervee, sergei.devadze, jaan.raik}@ati.ttu.ee, min@elin.ttu.ee*

**ABSTRACT:** We propose a new methodology for Built-In Self-Test (BIST) where contrary to the traditional scan-path based Logic BIST, the proposed solution for test generation does not need any additional hardware, and will not have any impact on the working performance of the system. A class of digital systems organized as pipe-lined signal processing architectures is targeted. The on-line generated signal data used for processing in the system serve as test pattern sources. Testing under normal working conditions and with typically processed data, allows exercising of the system on-line and at-speed, facilitating the detection of dynamic faults like delays and cross-talks to achieve high test quality. The proposed new self-test method is free from the negative aspect of over-testing, compared to the traditional Logic BIST approaches, and uses minimal amount of added hardware. Experimental research was based on the case study of specialized bio-signal processor architecture. The experiments showed promising results in reducing the cost of testing and achieving high fault coverage.

**Keywords:** pipe-lined signal processing architectures, built-in self-test, at-speed testing, design for testability

## 1 Introduction

The technology advancements impose new challenges to testing modern chips as device geometries shrink, and deep-submicron delay defects are becoming more and more important requiring more accurate dynamic tests than before [1]. Therefore testing of chips closer to real working conditions by so called at-speed test is becoming the must.

Increasing size and complexity of digital systems directly reflects in more demanding test generation and application strategies. The use of scan chains has proven to be often inadequate increasing the cost in terms of additional hardware and testing time [2], excessive power dissipation during test [3] and leading to yield loss because of over-testing [4].

A lot of research has been carried out to relieve the burden of external testers by introducing system self-test approaches like hardware-based Logic Built-in Self-Test (LBIST) which typically use Linear Feedback Shift Registers (LFSR) [5]. In LBIST, typical functions of external test equipment like test generation and response analysis are carried out on-chip, so that the tester should not handle high-speed signals externally and its role should remain only to send the test enable signals to the chip under test, and to receive the pass/fail signals. For example, scan-based and logic BIST solutions such as [6] relax the requirements on testers and considerably reduce the overall testing cost.

An important trend today is the at-speed test [7] having additional benefit of the ability to test circuits under conditions that are as close as possible to normal circuit operation. This factor has a direct impact on the number of chips that are found defective during system operation but still pass all manufacturing and functional tests. At-speed testing can be used for characterization and can also expedite test application time.

The question is whether a self-test sequence running in the system can adequately exercise its hardware components satisfying the targeted fault coverage requirements. Achieving the test quality target requires application of proper test sequences that is the focus of the current paper. It should also be pointed out that the quality of a test is measured not only by its fault coverage, but also by its code size (to be stored in the memory of the chip), hardware overhead, and by the test execution time.

The goal of the paper is to propose an approach which combines the ideas of traditional LBIST with at-speed testing to improve the test quality at less testing overhead and avoiding performance loss compared to the traditional self-test approaches. The feasibility and efficiency of the new method is demonstrated for a particular class of pipe-lined processing architectures which are easily adaptable for at-speed on-line self-testing by inherent functional sequences.

The rest of the paper is organized as follows. In Section 2, an overview about state-of-the-art is given. Section 3 presents the general idea of the proposed method, followed with the description of the evaluation bench in Section 4, which is needed for exploration of solutions for implementing the method. Section 5 presents the description of the representative case study, and in Section 6 the results of experimental research are discussed. Section 7 concludes the paper.

## 2 State-of-the-art of self-test techniques

In traditional LBIST, test pattern generation is mostly performed by Linear Feedback Shift Registers (LFSR) [5], cellular automata [8] or multifunctional registers like BILBO (Built-in Logic Block Observer) [5] to apply pseudorandom patterns to the Unit Under Test (UUT) and to analyse its output responses. However, many circuits contain random-pattern-resistant faults, which

limit the fault coverage that can be achieved with this approach.

One method to improve the fault coverage for LBIST is to modify the UUT by either inserting test points [4] or by redesigning it to improve the fault coverage [6]. The drawback of these techniques is that they generally add additional logic levels to the circuitry that can degrade system performance. Another possibility to improve the fault coverage is to use weighted pseudorandom test sequences [9]. The disadvantage of this approach is in the need of storing of the weight sets on chip, and also dedicated control logic is required to switch between weights, so the hardware overhead may become large.

A "mixed mode" approach, where deterministic patterns will be added to detect hard-to-test faults, has been developed in [10-13]. In [10] a technique based on reseeding LFSR was proposed that reduces the storage requirements. In [11], multi-polynomial LFSR for encoding a set of deterministic test cubes was introduced, and in [12] a technique called bit flipping for generating deterministic test cubes using BIST control logic was proposed. Further, in [13] a mixed-mode approach was presented in which deterministic test cubes are embedded in the pseudorandom sequence of bits itself.

Established BIST solutions use special hardware (typically LFSR) for test pattern generation (TPG) and test response evaluation (TRE) on chip [5], but this in general introduces significant area overhead and performance degradation. To overcome these problems, specialized methods were proposed which exploit specific functional units such as arithmetic units for on-chip test pattern generation [14, 16], which may afford to reach similar fault coverage like traditional LFSR-s. These methods are called Arithmetic BIST (ABIST), since they essentially adopt the additive congruential generation scheme of pseudo-random numbers [17].

In [18,19], a mixed-mode or hybrid BIST approach was proposed, where a test set is assembled from two parts, from pseudorandom test patterns that are generated on-line, and deterministic test patterns that are generated off-line and stored in the system. A combination of both test sources in an optimized fashion allowed improving the traditional LBIST in targeting hard-to-test faults. A similar approach called Hybrid Functional BIST (HyFBIST), where instead of LBIST the inherent functional sequences were used, was proposed in [20,21] for testing digital systems, and particularly micro-programmed data-paths.

In this paper we generalize and combine the ideas of using inherent functional blocks for test generation [14,15] and the inherent working sequences produced by the UUT itself for self-testing purposes. We propose an overall functional self-test concept for pipelined architectures where the working sequences are produced on the primary inputs of the system and the internal signals are monitored in selected test-points by Multiple Input Signature Analysers (MISR). We propose a systematic procedure for selecting the test-points to achieve the best overall fault coverage at minimum testing overhead and cost.

To our knowledge, the usage of digital representation of analog signal sequences as a functional test for testing digital circuits (signal processing architectures) is investigated in our paper the first time. Main idea is to take the input data, which is close to what the circuit-under-test would most probably have during its normal operation and apply this data as an at-speed test. In our case this input data is digital representation of the sine signal. It will be shown in results, that such a signal could yield better fault coverage in comparison to traditional pseudo-random LFSR sequence. This can also be considered as one step further compared to the arithmetic BIST (ABIST), since the source for the first stage of UUT is stimulated using more complicated equation (sine wave), than traditionally used in ABIST. The next stages of the UUT can be considered as test generators similar to ABIST. The Functional test strategies (e.g. software based self-test) used for example in microprocessors, are traditionally using dedicated software test routines, which have to be stored in the memory. In our case, there is no need to store in the memory such test routines or other test data.

## 3 General description of the method

Consider a digital system as a network of sub-circuits (blocks) where all the blocks may play simultaneously two roles: on one hand, each block will be itself UUT, and on the other hand, it will serve as the test pattern generator for the subsequent blocks it is feeding. As the overall test source, selected input working sequences (as functional test) will be used.

Two main problems arise: (1) how to find the best functional test sequences, and (2) how to find the minimal set of test-points for monitoring to achieve the highest fault coverage of testing.

In some cases, the first problem can be solved straightforwardly like in the instruction set architectures or in signal processing units. In the first case, the instructions can be exercised one by one where the problem recedes to finding only proper data (operands) as test patterns [22-23]. In case of signal processing units, the analog signals to be processed can be used as candidates for exploiting in testing purposes as well.

We are investigating the possibility of using given digital representation of analog signals as stimuli for testing signal processors. The idea is similar to random (LFSR based) testing where the critical point is analysis of the test quality as the function of test length.

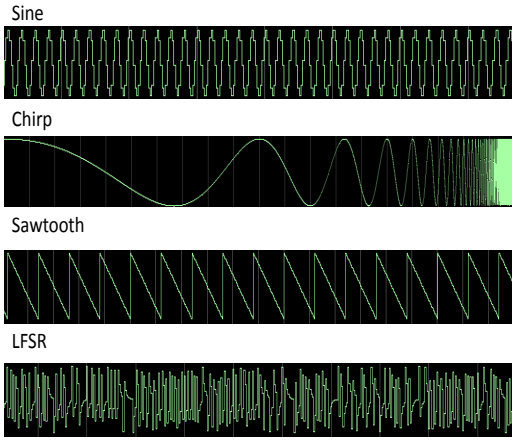



**Fig.1**. *Signals used for measuring bio-impedance*

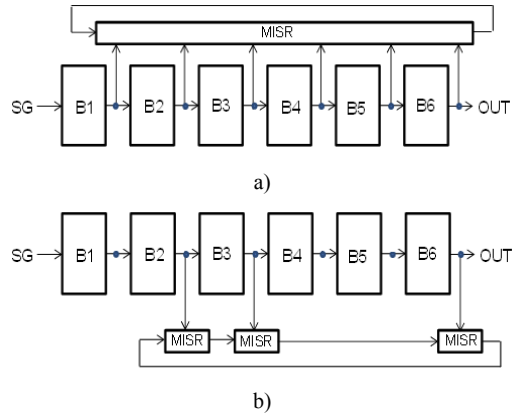



**Fig.2**. *Monitoring of the pipe-lined signal processing unit*.

For example, in bio-impedance spectroscopy, for measuring the bio-impedance typically the following signals are generated and processed as shown in Fig.1: *sine [28]* and *chirp [29]*. These signal sequences may be used as well in the role of stimuli (i.e., functional test sequences) for self-testing purposes for the same signal processor itself. The quality of the listed signals as test stimuli can be compared with popular *saw-tooth* analog signal and pseudorandom LFSR sequences which are traditionally used in the logic BIST solutions. *Saw-tooth* is easy to generate digitally; this is the reason why it is widely used in signal generation and processing. It can also be thought of as an additive generator of exhaustive patterns.

The second problem of selecting test-points for monitoring the test process depends how well can the faults in different blocks be detected by the given functional test sequence.

In Fig.2, an example of a pipe-lined signal processing unit is given which is partitioned into 6 blocks.

Two solutions are demonstrated for monitoring the behaviour of the circuit with MISRs. The solution in Fig.2a shows the case where all blocks are monitored whereas in the solution depicted in Fig.2b, only three MISR are used: the first is monitoring the behaviour of blocks B1 and B2 as a whole, the second MISR is monitoring solely the block B3, and the third MISR is monitoring the blocks B4, B5 and B6 as a whole.

The task of partitioning of the whole system into UUT blocks has the goal to find the highest fault detection coverage for the given functional test by achieving well-balanced testability at the minimum number of monitoring points equipped with MISR

To find the minimum hardware overhead we propose the following method for selecting test-points:

- Put MISR on the primary output of the circuit, and find the fault coverage (FC) for the given test sequence.
- If FC is sufficient, then the problem is solved.
- Partition the circuit into a set of *n* blocks (each with its own MISR). Find FC for each block as a UUT.
- Continue the partitioning of the blocks with low FC until the total FC will be sufficient.
- Integrate the consecutive blocks with high FCs into UUTs (with a single joint MISR in the output of the composite block) to minimize the number of MISR, so that the total FC of the system remains sufficient high.

The described method is illustrated in Fig.3. Please note that the partitioning solutions can be found in different ways, e.g. dictated by an inherent structure (network of registers and combinational blocks), using any ad hoc method in a style of "trials and errors" or using more sophisticated analysis methods. This task should be regarded as a separate problem, not discussed in the paper.

An example of merging two blocks with high fault detection coverage according to Step 4 of the described procedure and further partitioning of a block with low fault detection coverage into smaller blocks, is illustrated in Fig.4. In the new composite block, the evaluation of the fault coverage should be carried out again separately for all the three parts, to find out if some of them can be merged, or if there is any of them with low fault coverage, which should be further partitioned.

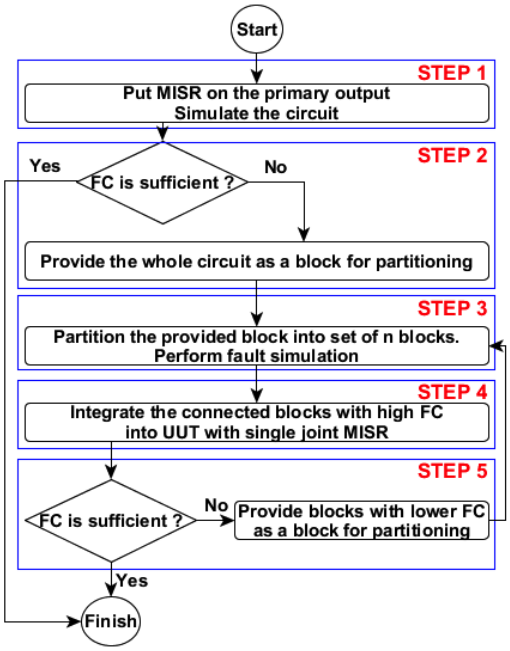



**Fig.3**. *General procedure for minimization of the number of observation points*

In case when the fault coverage will not satisfy either globally for the whole circuit or for particular blocks as UUTs, either the better functional test sequences should be found, or different methods, similar to the ones for improving LBIST described in Section 2, may be used.

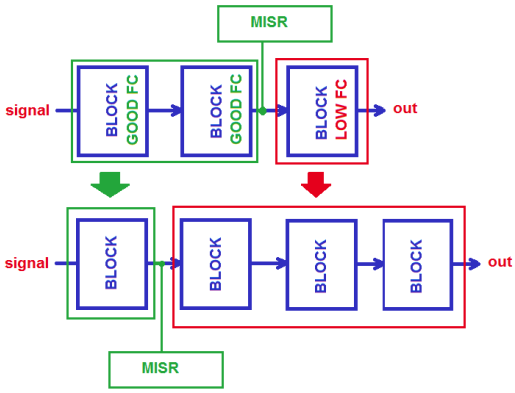



**Fig. 4.** *Example of merging and splitting the blocks in UUT with high and low fault detection coverage*

The described method of inserting MISR facilitates the idea of LBIST strategy except using instead of dedicated test generators of LBIST the inherent test functionality of existing hardware in the system. The method affords at-speed testing with no performance degradation and with little hardware overhead and reduced test cost.

The clock cycle based observation technique allows to avoid fault masking, and to achieve high fault coverage. The test response observation is carried out using built-in MISR as the only hardware overhead.

The proposed method has several advantages compared to the state-of-the-art scan-path based LBIST methods:

(1) no hardware test pattern generators, and no scan-path for shifting in external test patterns are needed, which results in smaller overhead;
(2) compared to LBIST, the typical drawback of over-testing related to LBIST is avoided, since only functional working test patterns are used;
(3) testing is carried out in the normal working clock-rate which guarantees at-speed exercising the whole circuit.

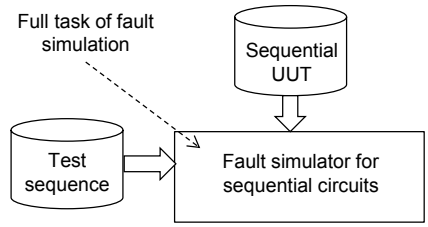



**Fig.5.** *Fault simulation in sequential circuits*

The target of this Section was to describe the main principles of redesign for better testability of the given UUT. The goal was not to develop exact algorithm or tool for exploring automatically the whole space of solutions, which would be infeasible. The designer has a possibility to remove or insert MISRs in the design and to evaluate the test quality by using the fault simulation environment described in Section 4. He has also the possibility of changing the length of the test sequence to achieve higher fault coverages.

In the next Section we present a novel environment which supports very high speed in analysing the fault detection coverage in the blocks of UUT.

## 4 Fault simulation environment

To carry out the procedure of minimizing the number of test-points according to Algorithm in Fig.3, large number of fault simulation sessions is needed for evaluating the fault detection coverage in the blocks of different size and for different partitioning solutions for the given UUT. A simple scheme for fault simulation of a sequential circuit is depicted in Fig.5. The model of the circuit and the test sequence form the input data for the simulator that calculates the fault detection rate. The

faults are simulated in this case one by one. Such a single fault simulation is very slow. On the other hand, faster methods for fault simulation, such as deductive or critical path tracing based fault analysis, cannot be used for sequential circuits; they are applicable only for combinational circuits.

To overcome the difficulties of fault simulation in sequential circuits we propose a special approach to escape from the dependency on feedback loops. Assume, the full sequential circuit (or a sequential block as a part of it) can be presented as a set of combinational sub-circuits each of them having a MISR in the output. By logic simulation of the test sequence for all sub-circuits, the input sequences are calculated (are fixed during the logic simulation). All the combinational sub-circuits can be fault simulated now independently, because each of them has MISR, which detects the faults in the related sub-circuit. If we cannot partition the circuit in such a way, we have to use a traditional slow fault simulator for sequential circuits.

To cope with the problem of slow fault simulation in sequential circuits, we have developed a novel environment in which the fault simulation has to be carried out only in the combinational parts of the UUT. The new fault simulation environment is depicted in Fig.6.

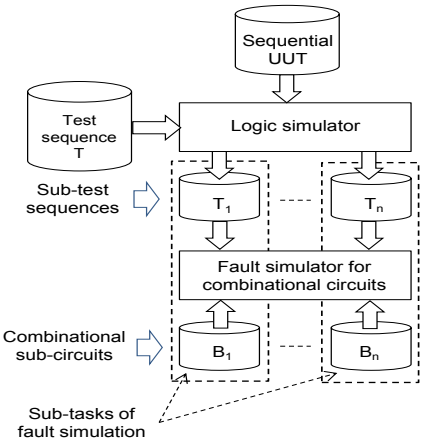



**Fig.6.** *Transforming sequential fault simulation into sub-tasks of combinational fault simulation*

The fault simulation in this environment is carried out in the following flow:

(1) In each current step of the Algorithm in Fig.3, the UUT is partitioned into a set of blocks $S = S_C \cup S_S$ where $S_C$ is a subset of combinational blocks and $S_S$ is a subset of sequential blocks.

(2) The UUT is simulated for the whole test sequence $T$, and for each block $B_i \in S$, the whole local subsequence $T_i$ at the input of $B_i$, caused by $T$ will be collected and stored. The subsequence $T_i$ will be regarded thereafter as the sub-test sequence for the block $B_i$ generated on-line by the test sequence $T$.

(3) All the combinational blocks $B_i \in S_C$, will be fault simulated for the local sub-test sequences $T_i$ with the fast fault simulator for combinational circuits as shown in Fig.6.

(4) All the sequential blocks $B_j \in S_S$, have to be fault simulated for the local sub-test sequences $T_j$ with the slow fault simulator for sequential circuits in this environment according to scheme Fig.5.

For fault simulation of combinational circuits we have developed a very fast fault simulator that implements a method of exact parallel critical path tracing, which has higher speed than currently used commercially available professional fault simulators have [24].

The high speed in our simulator is achieved by reasoning the faults along signal paths in the circuit for $N$ test vectors in parallel, where $N$ is the number of bits in the computer word. The simulator runs in two sessions through the whole circuit. The first session is carried out only once for all the test vectors to be simulated. The goal of this session is to create a compact computing model for further fault reasoning which consists of a sequence of Boolean formulas. Since the formulas are Boolean, they can be processed in parallel. The second session is to calculate the detected faults for packages of N test vectors in parallel using the computing model created in the first session.

We included this simulator into the fault simulation environment in Fig.6, where it will be used for simulating faults in the blocks $B_i \in S_C$, block by block.

Unfortunately, the simulator cannot be used for calculating the fault coverage for the sequential blocks $B_j \in S_S$. Currently, we are working on the problem, how to use the simulator proposed in [24] for estimating fault coverage in sequential circuits as well.

In the next Section, a case study will be discussed where we investigated the feasibility of the proposed method of at-speed self-testing in a pipe-lined signal pre-processor of a family of pre-processors with different architectures developed for analysis of electrical bio-impedance signals [new, 25, 26]. The results of fault simulation for the whole family of 8 processors (column 1) are presented in Table 1, and they allow comparing the performance of the two simulation schemes depicted in Fig.5 and Fig.6.

Column 2 describes the time in seconds for logic simulation of the sequence of 10 000 vectors on these processors given by their behaviour VHDL descriptions. The columns 3 - 6 describe fault simulation experiments according to Fig.6 on the same sequence of 10000 vectors. Two levels of fault simulation are compared – gate-level and macro-level, where each macro represents a fan-out-free region (a gate-level sub-circuit) in a

simulated combinational block of the given UUT. Only Stuck-at-Faults (SAF) were simulated. However, to save the time, only the correct behaviour was considered during behaviour level simulation (column 2).

| Circuit | Beh. level logic simulationn, [sec] [Fig.5] | Fault simulation (Fig.6) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Macro level | | | Gate level | | |
| | | # of faults | Sim-n time, [sec] | Speedup | # of faults | Simul. time, [sec] | Speedup |
| 8a | 0.155 | 66328 | 14.0 | 734 | 112034 | 30.0 | 579 |
| 8b | 0.152 | 50206 | 12.1 | 631 | 83940 | 24.7 | 517 |
| 8be | 0.168 | 55270 | 28.3 | 328 | 99330 | 62.1 | 269 |
| 8bk | 0.159 | 49938 | 11.6 | 684 | 86878 | 25.2 | 548 |
| 8bs | 0.154 | 56444 | 65.2 | 133 | 100820 | 173.4 | 90 |
| 8c | 0.159 | 73182 | 16.3 | 714 | 122386 | 35.9 | 542 |
| 8d | 0.161 | 71730 | 17.0 | 679 | 123012 | 35.5 | 558 |
| 8de | 0.164 | 75840 | 34.8 | 357 | 136876 | 81.3 | 276 |

**Table 1.** *Comparison of two fault simulation approaches*

To compare the two fault simulation approaches presented in Fig.5 and Fig.6 on the basis of Table 1, let consider the results for the processor architecture 8a (the 1st row). For simulating 112034 gate-level SAF faults using parallel critical path tracing in the environment in Fig.6, we need 30 s. Assume now very optimistically that for single fault simulation of sequential circuits in the UUT at the behaviour level we would need the same time as for simulation of the correct circuit, i.e. 0.155 s. Then, to simulate 112034 faults in the sequentially presented gate-level UUT, we would need 17365 s, or about 5 hours. Hence, the gain in speed for this particular UUT will be not less than 580 times. In fact, it will be even more, since the gate-level simulation would be much slower than the behaviour level simulation.

In order to produce the results in Table I used desktop class intel I7-930 @ 2,80GHz 4-core processor running Windows 7 operating system with 6GB of physical RAM. The circuit was simulated by ModelSim SE ver.6.5c. The speedup values were calculated in respect to theoretically assumed speed of sequential fault simulation computed as multiple of column two and respective column for macro- and gate-level number of faults.

Note, the main idea of such a powerful fault simulation, based on transforming sequential fault simulation task into a set of combinational fault simulation sub-tasks is directly related to the goal we have in this analysis. And the idea is as well closely tailored in the method of at-speed testing that we are evaluating. The goal of fault simulation is in our case to evaluate the fault detection coverage, not fault diagnosis. In other word, we are not interested in creation of an exact fault table. As we have on the outputs of simulated blocks signature analysers, it will be sufficient during testing to fix on the inputs of the block correctly only the first erroneous vector affected by the fault. As the result, the method is not sensitive to the possible mismatches of the subsequent input vectors of the fault block with those collected during simulation of the correct UUT.

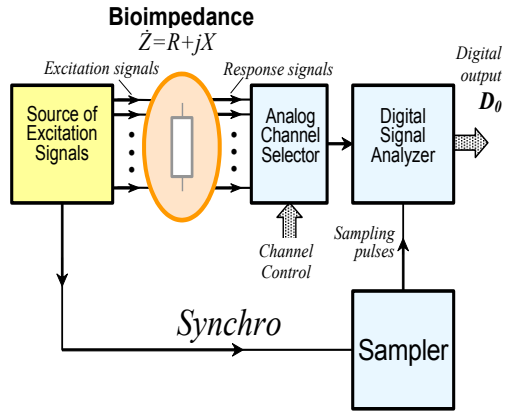



**Fig. 7.** *The architecture 8a of Signal Analyser.*

Let us summarize the main idea of the Section. We use our fast fault simulator [24], which is not simulating faults one by one like in the traditional fault simulators for sequential circuits, rather it calculates by a single run all the faults detected in the combinational sub-circuits by a bunch of patterns (we do the reasoning of all faults in the sub-circuits in parallel for many patterns). The confusion may arise now because the fault reasoning is carried out for input patterns, which were collected from the behavior of the correct circuit. This means that if there was a fault, which produces an erroneous output pattern, then the next input pattern will be as well erroneous (because of the feedback loop), which means in turn that the results of fault reasoning of all subsequent patterns will be as well wrong. But, on the other hand, this is not any more important, because the first erroneous pattern in the input sequence of the sub-circuit will be fixed already by MISR as an error, and this will be sufficient for fault detection in the end of the test (with the accuracy determined by the probability of signature aliasing). Generating fault tables and fault diagnosis of course is not possible, but this is not the purpose of our paper.

To our knowledge, such an approach of handling feedback loops during fault simulation in sequential circuits has been proposed the first time.

## 5 Case study: signal processing unit as UUT

To investigate the feasibility of the method in the sense of achieving sufficient fault coverage in real cases, we carried out experimental test research with a digital signal pre-processor unit developed for industrial purposes for measuring electrical bio-impedance [25, 26, 27]. From the family of processors, discussed in the previous Section we selected the processor with architecture 8a.

The selected unit under test (UUT) is a bio-impedance signal analyser that implements a simplified signal processing algorithm [25, 27]. The architecture of the signal analyser circuit is depicted on Fig. 7. A typical digital solution is that the response voltage is digitized in an analog-to-digital converter (ADC) into a uniformly sampled train of digital data, which is then processed numerically in a digital signal processing (DSP) unit, often using the Discrete Fourier Transform (DFT). Because the whole signal path from the generation of the set of excitation signals to the analog-to-digital conversion procedure and data analysis is synchronous by design, optimized signal processing methods can be applied. Using of sampling, which is synchronous to the known excitation waveform, enables to use a simplified but much faster signal processing than the Fourier Transformation is. When sampling the response signal uniformly with intervals $t = T/4$, where $T$ is a period of the signal, the following simple mathematics is valid [25][30]:

(1) the direct current component DC can be determined as

$$DC = (Re^{+} + Re^{-})/2 \quad \text{or} \quad DC = (Im^{+} + Im^{-})/2$$

(2) the real $Re$ and imaginary $Im$ parts of the phasor of complex bio-impedance Z is determined as

$$Re = (Re^{+} - Re^{-})/2, \quad \text{and} \quad Im = (Im^{+} - Im^{-})/2$$

The mentioned signal analyser is a part of the developed digital multichannel bio-impedance analyser (DMBA) [27], depicted in Fig. 7. The Source of Excitation Signals generates digital waveforms that are converted by Digital Analog Converter (DAC, not shown) into analog signals, sent through tissue, collected by Analog Channel Sellector and sampled converted by ADC (not shown) back to digital form. The Sampler is used to synchronize the signal source and ADC (input of the Digital Analyzer, not shown). This sampled digital signal is processed by the analysis unit. For self-test purposes, the analog part - DAC, tissue and ADC - are skipped and the output of the Excitation Signal Generator is fed directly to the digital input of the Digital Signal Analyzer, which is shown in Fig.9.

For the test purposes of the circuit, the sampler is implemented as 80MHz clock signal and excitation signal generator along with body and analog part was exchanged with signal generator of particular type. Fig.9. gives visual representation of it. As it can be seen, the circuit has a pipe-line structure in order to be implemented in a low-cost FPGA. The signal is first sampled into the input buffer. On the next stage the signals are distributed to the particular registers of the 8 different channels. The sampling is performed on channed-after-channel basis. Every sample out of 4 samples taken per channel is saved into its corresponding 16-bit register. On the next stage Real and Imaginary and Direct current components are computed with adders and subtractors, using equations (1) and (2). On the next pipeline stage the computed components are integrated using adders and saved into 32-bit registers, called output buffers. Integration is made over a 1 millisecond period. After that the values of the output buffers are transferred to the output register of the anayser.

The architecture of the analyzer (see Fig. 8) was defined by the used technology - low-cost FPGA-s - that defined the used 80 MHz sampling frequency. This itself was defined by the need to have 10-20 MHz excitation signals with 4 or 8 sampling points per period [27]. The use of single input channel (ADC output) and sorter to reduce aliasing effect [25], defines the need for two first buffer stages. The two last buffer stages are defined by the need to accumulate the collected data over 1 ms period, to buffer it and to transmit [27]. The intermediate part - subtracter/adder and accumulator - can be implemented, in principle, as a single stage. However, when using FPGA-s, the extra pipeline stage actually makes the design not only faster (because of the shorter combinational paths) but also smaller - every output bit of an adder has a flip-flop anyway and the use of them

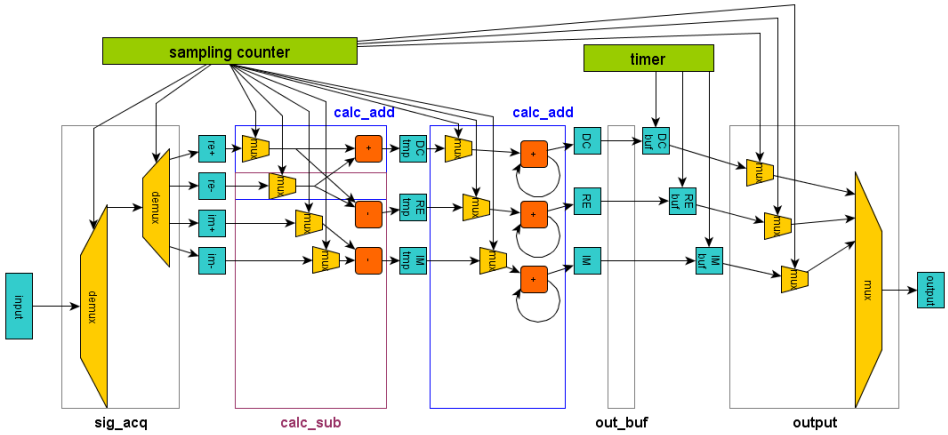



**Fig.8.** *A block diagram of one channel of DMBA*[25]*. Registers are blue, MUXes – yellow, computation - red, control – green.*

makes routing problem for the design tools easier. The same applies for the potential reuse of functional units and registers that would essentially add additional multiplexers to the design - the internal structure of FPGA-s is best suited for pipelined data-stream oriented applications. This was also the case with the other implementations of the same processing unit with the different degree of reuse [25]. All the MUXes in the circuit are used for switching the channels (there are 8 channels in the DMBA), except the last one, which is switching every channel result to a single output.

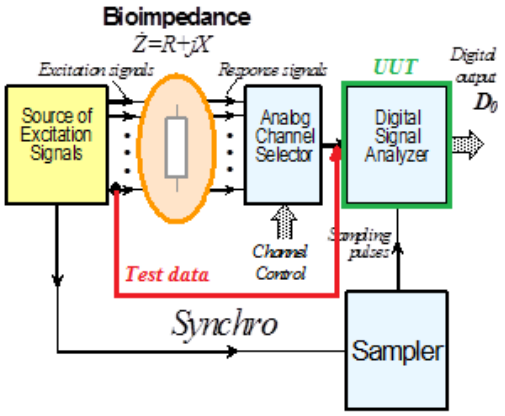



**Fig. 9.** *Testbench for the case study*

We investigated four types of signal generators for using as test sources to provide input signals to the channels of the analyser: *sine*, *chirp*, *saw-tooth*, LFSR. These can be see on Fig.1. All the channels get the same signal, so that we can test each channel equally to each other. All the generators are implemented for the simulation environment in VHDL.

(1) The *sine* signal generator is using floating point arithmetics and *sin*() function of the VHDL math library. It can take amplitude, phase and frequency as a parameters to produce the corresponding *sine* wave. During the experiments the amplitude was set to 15 bits, taking into account 1 sign bit and 16-bit wide input of the analyser. The phase was set to 90 degrees in order to produce the input signal from the upper part of the wave. This was done because the signal would produce more unique values in less time, because it covers all the values from top to the bottom in half-period. It was useful to check whether the test sequences of small length could produce meaningful results. The frequency was modified during the experiment in order to detect the better signal for testing this device.

(2) The *chirp* generator takes as parameters start and stop frequency periods as well as number of samples in which frequency should change from start to stop frequency. The *chirp* generator changes the frequency every sample it produces. The amplitude remained 15bits + 1 sign bit and phase remained 90 degrees. During the experiments we have changed the length of the *chirp* signal – number of samples from start to end frequency.

(3) *Saw-tooth* signal is implemented as a counter. The parameter it takes is a period of the signal. The generator produces equally spaced samples of the *saw-tooth* signal of this period. The amplitude is 15bits+1 sign bit.

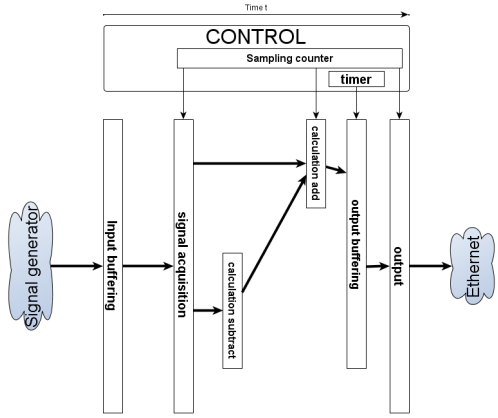



**Fig. 10.** *Equivalent circuit for the Signal Analyser in Fig.8.*

(4) LFSR signal generator is implemented as 16-bit linear feedback shift register. The seed is taken so that it goes through all the 65535 possible values except 0. The size of the LFSR was chosen in accordance to the input width of the signal analyser under test.

## 6 Experimental results

Experiments were carried out for Signal Analyser (architecture 8a) in Fig.9, presented as equivalent circuit with highlighted pipe-lined tracks in Fig.10. As the result of the experimental research according to method in Fig.3, the circuit was finally partitioned into 7 blocks as separate UUTs which are characterized in Table 2.

| No | Name of the block | Number of faults | Number of inputs | Number of outputs |
|---|---|---|---|---|
| 1 | *calc_add* | 69544 | 1431 | 896 |
| 2 | *calc_sub* | 18588 | 791 | 256 |
| 3 | *in_buf* | 98 | 17 | 16 |
| 4 | *out_buf* | 14750 | 1554 | 769 |
| 5 | *out* | 7480 | 709 | 64 |
| 6 | *sig_acq* | 8560 | 538 | 520 |
| 7 | *timer* | 512 | 18 | 17 |
| Total | | 119532 | 2528(5058) | 2538 |

**Table 2.** *Characteristics of the blocks in Fig.9*

We calculated the fault coverage for all the 7 blocks as well as the total fault coverage for four different types of signals: sine, chirp, saw-tooth and LFSR. The number of

test patterns used for this simulation is 1000000. The results of the experimental research in percentage of fault coverage for all the different blocks are presented in Table 3 and as the bar diagram in Fig. 11. Blocks *timer* and *sampling* represent control logic of the circuit. These are well tested, because their memory cells are completely covered by MISRs. The reason, why the coverage is not 100% is that we didn't simulated the reset logic of the circuit.

As we see, the best results in average for all the blocks were achieved for the input signal *sine* where the fault coverage was 98.20%. The lowest total fault coverage 75.99% was registered for the signal type *saw-tooth*.

| No | Name of the block | Input signal types | | | |
|---|---|---|---|---|---|
| | | *Sine, %* | *chirp, %* | *saw-tooth, %* | *LFSR, %* |
| 1 | *calc_add* | 97.37 | 94.86 | 76.80 | 95.71 |
| 2 | *calc_sub* | 98.85 | 99.20 | 64.90 | 99.20 |
| 3 | *in_buf* | 82.65 | 82.65 | 82.65 | 82.65 |
| 4 | *out_buf* | 99.88 | 99.86 | 74.74 | 99.86 |
| 5 | *out* | 99.14 | 99.06 | 78.66 | 99.14 |
| 6 | *sig_acq* | 95.63 | 95.63 | 95.63 | 95.63 |
| 7 | *timer* | 94.14 | 94.14 | 94.14 | 94.14 |
| 8 | *sampling* | 95.62 | 95.62 | 95.62 | 95.62 |
| Total | | 98.20 | 96.68 | 75.99 | 97.21 |

**Table 3**. *Results of fault coverage experiments*

Considering the distribution of fault coverage among different blocks we see that the lowest test quality is mapped to the block **in_buf**. Hence, for this block the improvement of the testability by any of the methods referenced above in Sections 2 and 3 can be foreseen (this task was not the goal of this case study paper). However, since the block **in_buf** is rather small (characterized by only 98 faults), the improvement of its testability will not lead to considerable increase in the total fault coverage of the whole circuit.

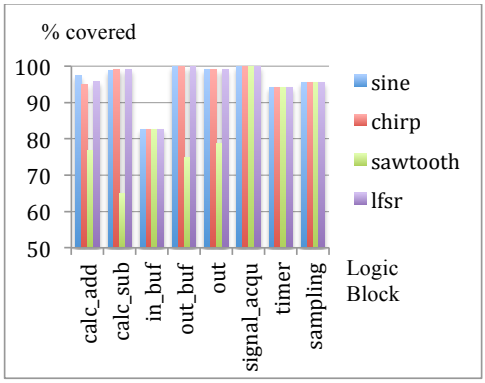



**Fig 11**. *Distribution of fault coverage in the circuit*

Since the cost of testing depends on the time used for carrying out the self-test procedure, we investigated how the fault coverage will depend on the test length measured in the number of test patterns. The results are shown as the graphics for the different four signal types in Fig. 12.

The most cost effective would be the LFSR based self-test sequence where the fault coverage around 90% will be achieved already after 80 000 test patterns (clock cycles) whereas the sine signal based and chirp signal based tests achieve only about 85% and 80% fault coverage, respectively, at the same test length. When doubling, however, the test length, the sine based and LFSR based tests become equal at the 95% fault coverage. Especially sensitive to the length of the test is the chirp signal based test sequence.

We compared the test quality achieved by the proposed method with traditional scan-path (SP) techniques both for using LFSR pseudorandom and deterministic test sequences. The results are presented in Table 4.

As we see from Table 4, the fault coverage is nearly the same for all the methods compared. However, to get the same fault coverage as with the proposed method, the test length of the scan path & LFSR based approach should be even twice bigger compared to the proposed method. To calculate the testing time cost in clock cycles, the test length for both referenced scan-path based methods should be multiplied by the length of the scan path which is equal to 2528 bits (the total number of inputs of all the tested blocks in the given circuit).

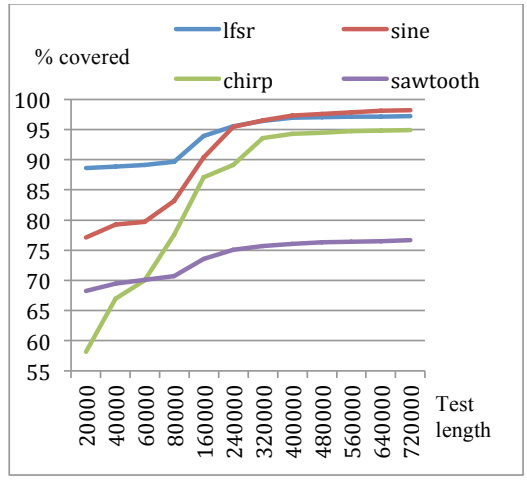



***Fig 12***. *Dependence of the fault coverage on test length*

For the proposed method, the testing time in number of clocks is equal to the test length. Hence, we can conclude that the time cost of the proposed method is about 5-7 times cheaper than the SP & deterministic approach and more than 2500 times cheaper than SP &

LFSR at the same fault coverage (in the latter case the single scan-path was assumed).

| Method | Fault cover % | Test length (TL) | Testing time (clock cycles) |
|---|---|---|---|
| Proposed | 97.78 | 500000 | **5** *$10^5$ |
| SP & LFSR | 96.82 | 500000 | 12640 *$10^5$ |
| Proposed | 98.20 | 1000000 | **10** *$10^5$ |
| SP & LFSR | 98.73 | 1000000 | 25280 *$10^5$ |
| SP & deterministic | 98.69 | 1364 | **34** *$10^5$ |

**Table 4.** *Comparison of different methods*

## 7 Conclusions

We introduced a new approach to self-testing of digital systems with pipe-lined architectures using inherent functionalities of systems with capability to produce internal self-test sequences. The added value of using inherent functional self-test sequences is the higher test quality explained by on-line at-speed testing. The approach does not need to store high volume test data in the system memory. Additional hardware is as well not needed for on-line test pattern generation as in the case of traditional LBIST. The only needed additional test hardware is related to using MISR for monitoring the test responses. To minimize the needed additional MISR hardware overhead, an original algorithm for selecting test-points was developed. As the result of avoiding artificial embedded test pattern generators like in case of LBIST, and of using only normal working sequences for test purposes, the danger of over-testing and the related yield loss are removed.

To cope with the problem of very slow fault simulation in sequential circuits, needed for exploration and comparison of different self-test solutions we developed a novel evaluation environment where the time consuming sequential fault simulation task can be transferred into a set of combinational fault simulation sub-tasks. Experiments demonstrated the gain in evaluation speed more than 580 times without losing any accuracy in fault coverage calculation.

To investigate the feasibility of the method to achieve high fault coverage, we carried out experimental research with a digital Signal Analyser unit as a case study, which was developed for industrial purposes for measuring electrical bio-impedance.

The goals of the experiments were twofold: (1) to select the best type of input signal for testing purposes from a set of signals typically used for processing in the given Signal Analyser, and (2) to compare the new method with traditional scan path based testing methods**.**

Experimental research showed that the best testing capability has the *sine* signal (with fault coverage of 98.2%) compared to the LFSR based pseudorandom (97.2%) and *chirp* (96.7%) signals at the same test length. The worse testing capability has the *saw-tooth* type signal (76%). The fault coverage achieved by the *sine* signal was 98.2%, which is nearly the same compared to the traditional scan-path pseudorandom (98.7%) and deterministic (98.7%) test approaches. The gain in testing time cost was 3-7 times compared to the deterministic and more than 2500 times compared to the pseudorandom single scan-path based approach.

Acknowledgements: The work was supported in part by EU FP7 STREP project BASTION, Estonian ICT project FUSETEST, by EU through the European Structural and Regional Development Funds, by the Estonian Doctoral School in Information and Communication Technology and by the IT Academy Program of Information Technology Foundation for Education.

## References

[1] T. Mak, S. Krstic, K.-T. Cheng, L.-C. Wang. New challenges in delay testing of nanometer, multi-gigahertz designs. IEEE Design & Test of Computers, 21(3), 2004, 241-248.

[2] L. Bushard, N. Chelstrom, S. Ferguson, B. Keller. DFT of the Cell Processor and its Impact on EDA Test Software. In IEEE Asian Test Symposium, 2006, pp. 369-374.

[3] S. Wang, S. Gupta. ATPG for heat dissipation minimization during scan testing. In ACM IEEE Design Automation Conference, 1997, pp. 614-619.

[4] L. Chen, S. Ravi, A. Raghunathan, S. Dey. A Scalable Software-Based Self-Test Methodology for Programmable Processors. In IEEE/ACM Design Automation Confernce, 2003, pp. 548-553.

[5] L.-T. Wang, C.-W. Wu, X. Wen. VLSI test principles and architectures. Morgan Kaufmann, 2006.

[6] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, J. Rajski.. Logic BIST for large industrial designs. Proc. IEEE Int. Test Conf., pp. 358–367, 1999.

[7] B. Nadeau-Dostie. Design For At-Speed Test, Diagnosis and Measurement. Kluwer Academic Publishers, 2002.

[8] P.D. Hortensius, R.D. McLeod, B.W. Podaima. Cellular automata circuits for BIST. IBM J of Research and Development. Vol.34, No.2.3, pp.389-405, 1990.

[9] M.F. AlShaibi, Ch. Kime. MFBIST: A BIST method for random pattern resistant circuits. Proc. ITC, Oct. 1996, pp.176-185.

[10] B. Koenemann. LFSR-coded test patterns for scan designs. Proc. European Test Conf., Mar. 1991, pp.237-242.

[11] S. Hellebrand, J. Rajski, S. Tarnick, B. Courtois, S.Venkataraman. Built-in test for circuits with scan based on reseeding of ,ulti-polynomial linear feedback shift registers. IEEE Trans. On Comput. Vol. 44, pp.223-233, Feb. 1995.

[12] H.-J. Wunderlich, G. Kiefer. Bit flipping BIST. Proc. ICCAD, Nov. 1996, pp.337-343.

[13] N.A. Touba, E.J. McCluskey. Bit-fixing in pseudorandom sequences for scan BIST. IEEE Trans. on CAD of IC and Systems, Vol.20, No.4, Apr.2001.

[14] R. Dorsch, H-J. Wunderlich. Accumulator based deterministic BIST. ITC, 1998, Washington D.C.

[15] J. Rajski, J. Tyszer. Arithmetic BIST in embedded systems, Prentice-Hall, N J (1998).

[16] I. Voyiatzis, D. Gizopoulos, A. Paschalis. Accumulator-based test generation for robust sequential fault testing in DSP cores in near-optimal time. IEEE Trans. on VLSI Systems, Vol.13, No.9, Sept., 2005, pp.1079-1086.

[17] D.E. Knuth. The art of computer programming. Vol.2, Addison-Wesley, 1981.

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

[19] G. Jervan, P. Eles, Z. Peng, R. Ubar, M. Jenihhin. Hybrid BIST Time Minimization for Core-Based Systems with STUMPS Architecture. 18th Int. Symposium on Defect and Fault Tolerance in VLSI Systems. Cambridge, MA, USA, November 3-5, 2003.

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

[21] N. Mazurova, J. Smahtina, R. Ubar. Hybrid Functional BIST for Digital Systems. Proc. of the 9th Biennial Baltic Electronics Conference, Oct. 3-6, 2004, Tallinn, pp.205-208.

[22] D. Gizopoulos et al. Systematic software-based self-test for pipelined processors. IEEE Trans. on VLSI Systems, Vol. 16, No.11, Nov. 2008, pp.1441-1453.

[23] R. Ubar, V. Indus, O. Kalmend, T.Evartson. Functional Built-In Self-Test for Processor Cores in SoC. The 30th IEEE NORCHIP Conference, Copenhagen, Denmark, Nov. 12-14, 2012.

[24] R. Ubar, S. Devadze, J. Raik, A. Jutman, "Parallel X-Fault Simulation with Critical Path Tracing Technique". DATE, 2010.

[25] P. Ellervee, P. Annus, M. Min. High Speed Data Preprocessing for Bioimpedance Measurements: Architectural Exploration. NORCHIP, 2009.

[26] H. Kruus, R. Ubar, P. Ellervee, M. Brik, M. Gorev, M. Kruus, E. Orasson, V. Pesonen, P. Annus, M. Min, K. Meigas. A Benchmark Suite for Evaluating the Efficiency of Test Tools. Proc. of Baltic Electronics Conference, Tallinn, October 3-5, 2012.

[27] P. Annus, A. Kuusik, R. Land, O. Märtens, A. Ronk, “A Digital Multichannel Bioimpedance Analyser: Signal Processing Task and its Solution”. Instrumentation and Measurement Technology Conference (IMTC 2006), Sorrento, Italy, Apr. 2006.

[28] Min, M., Annus, P., Land, R., Paavle, T., Haldre, E., Ruus, R., "Bioimpedance Monitoring of Tissue Transplants",Instrumentation and Measurement Technology Conference Proceedings, 2007. pp: 1- 4

[29] Paavle, T., Min, M., Parve, T., "Using of chirp excitation for bioimpedance estimation: Theoretical aspects and modeling", 11th International Biennial Baltic Electronics Conference, 2008. pp: 325-328

[30] Min, M., Land, R., Martens, O., Parve, T., Ronk, A., "A sampling multichannel bioimpedance analyzer for tissue monitoring", 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2004. pp: 902-905

[31] J.Kõusaar, R.Ubar, S.Devadze, J.Raik. Critical Path Tracing based Simulation of Transition Delay Faults. The EUROMICRO Conference on Digital System Design – DSD, Verona, Italy, Aug. 27-29, p. 1-6.

# APPENDIX D

# Fault Simulation with Parallel Exact Critical Path Tracing in Multiple Core Environment

Maksim Gorev
Department of Computer Engineering
Tallinn University of Technology
Tallinn, Estonia
Email: maksim.gorev@ttu.ee

Raimund Ubar
Department of Computer Engineering
Tallinn University of Technology
Tallinn, Estonia
Email: raimund.ubar@ati.ttu.ee

Sergei Devadze
Department of Computer Engineering
Tallinn University of Technology
Tallinn, Estonia
Email: serega@pld.ttu.ee

***Abstract*—A novel fault simulation method is proposed, based on exact critical path tracing beyond the Fan-out-Free Regions (FFR) throughout the full circuit. The method exploits two types of parallelism: bit-level parallelism for multiple pattern reasoning, and distribution the fault reasoning process between different cores in a multi-core processor environment. To increase the speed and accuracy of fault simulation, compared with previous methods, a mixed level fault reasoning approach is developed, were the fan-out re-convergence is handled on the higher FFR network level, and the fault simulation inside of FFRs relies on the gate-level information. To allow a uniform and seamless fault reasoning, Structurally Synthesized BDDs (SSBDD) are used for modeling on both levels. Experimental research demonstrated very promising results in increasing the speed and scalability of the method.**

## I. Introduction

Fault simulation is one of the most important tasks in the digital circuit design and test flow. The efficiency of solving other tasks in this field like design for testability, test quality and dependability evaluation, test pattern generation, fault diagnosis relies heavily on the performance and speed of fault simulation. Such a dependence is growing especially in case of large circuits, and hence, the scalability of the fault simulation algorithms is decisive. Accelerating the fault simulation would consequently improve all the above-mentioned applications.

Parallel pattern single fault propagation (PPSFP) concept [1] has been widely used in combinational and full scan-path circuits for fault simulation. Many proposed fault simulation concepts incorporate PPSFP with other sophisticated techniques such as test detect [2], critical path tracing [3], [4], stem region [5] and dominator concept [4], [6]. These techniques have helped to reduce further simulation time. Another trend of fault simulation methods based on reasoning (deductive [7], concurrent [8] and differential simulation [9]) used to be very powerful since they allow to collect all detectable faults by a single run of the given test pattern. What they cannot do, is to produce reasoning for many test patterns in parallel.

The critical path tracing method [3], [4] eliminates explicit fault simulation for faults within Fan-out-Free Regions (FFR). A modified critical path tracing technique that excludes fault simulation for fan-out stems and includes a system of rules to check the exactness of critical path tracing beyond the FFRs, and which is linear in time, is proposed in [10]. However, the rule based strategy does not allow parallel analysis and rule check of many patterns simultaneously. This drawback was removed in [11] by introducing a novel concept of Parallel Pattern Exact Critical Path Tracing (PPECPT) which can be applied efficiently also beyond FFRs. In [12], the same method was extended from stuck-at faults (SAF) for a general class of X-faults. The main idea of the method was in compiling of a dedicated compact computing model through the circuit topology analysis, which allows exact critical path tracing throughout the full circuit and not only inside FFRs.

In this paper we propose a new PECPT method, where we implement two types of parallelism during fault simulation: (1) bit-level parallelism for multiple pattern reasoning, and (2) distributing the compiled computing model among a subset of different CPUs in a multi-core computing environment, so that each processor were responsible for parallel critical path tracing in a related particular sub-circuit area.

Another novelty of the paper is in developing of a mixed level fault reasoning approach, were the problems related to the fan-out re-convergence are handled on the higher FFR network level, using collapsed fault set, and the increased speed and accuracy in fault reasoning is achieved by fault reasoning inside FFRs using additional gate-level simulation data, without fault collapsing.

To speed-up simulation and improve the accuracy of fault reasoning compared with previous methods in [11], [12], we propose here a mixed level PPECPT method based on using of two types of Structurally Synthesized BDDs (SSBDD).

Since during a single run of parallel analysis of patterns throughout the circuit we process all the faults in the circuit, we can say that the approach we propose is exploiting concurrency in three dimensions: pattern dimension, fault dimension and computing model dimension, where the pattern and fault parallelism is utilized using each single CPU core, while computing model concurrency is achieved exploiting multiple CPUs. Compared to the traditional approaches which can use only pattern- and fault-parallelism in multi-CPU systems (at the bit and system level, respectively), such a new dimension addition gives further possibilities to speed up fault simulation in multi-processor systems.

The availability of parallel execution environments such as multiprocessor system on chips (MPSoCs), multicore processors and GPGPU devices provides a possibility for concurrent execution of the same algorithm for different data or for different parts of the same algorithms and the same data. This is done to utilize the available new hardware resources, as well as to speed up execution in comparison to uniprocessor system. In the landscape of fault simulation, where this paper is targeted, the growing size and complexity of digital circuits also requires speed up of available algorithms.

The rest of the paper is organized as following. In Section 2 we present the theoretical basics of exact fault simulation by parallel critical path tracing beyond the fanout stems. In Section 3 we present the basics of fault simulation using SSBDDs, and in Section 4 we propose a new method of parallel critical path tracing based on mixed level fault simulation with two types of SSBDDs. Section 5 describes the method of distributing the task between multiple cores of the processor, Section 6 describes the results of experimental research

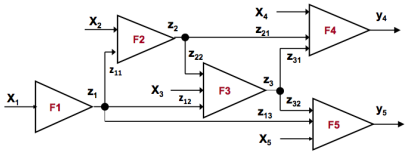



Fig. 1. Combinational circuit with 5 FFRs.

with related discussion, and Section 7 concludes the paper.

## II. Parallel pattern critical path fault tracing

Consider a combinational circuit as a network of FFRs, where each of them is represented as a Boolean function

$$y = F(x_1, x_2, \ldots, x_n) = F(X) \tag{1}$$

where $X = x_1, x_2, \ldots, x_n$ is the input vector of the FFR. Such a network of 5 FFRs is represented in Fig.1. Let $X_k$ denote the vector of input variables of the k-th FFR, $z_k$ denote the internal fan-out stem variables (outputs of FFRs) with $z_{kj}$ as fan-out branch variables for $z_k$ (inputs of FFRs) and y denote the output variables of the circuit.

The fault simulation can be processed as calculation of Boolean derivatives: if $\partial y/\partial x = 1$ then the fault is propagated from $x$ to $y$. This check can be performed in parallel for a set of test patterns. In order to extend the parallel critical path tracing beyond the fan-out free regions we use the concept of Boolean differentials [13].

Consider the full Boolean differential of the FFR $y = F(X)$ as

$$dy = y \oplus F((x_1 \oplus dx_1), \ldots, (x_n \oplus dx_n)) \tag{2}$$

Here, by $\partial x$ we denote the change of the value of $x$ because of the influence of a fault at $x$, and $\partial y = 1$ if some erroneous change of the values of arguments of the function (2) causes the change of the value of $y$, otherwise $\partial y = 0$.

In [11] we have shown that from the expression (2) the following relationship can be derived:

$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}dx), \ldots, (x_n \oplus \frac{\partial x_n}{\partial x}dx)) \tag{3}$$

For example, the fault at $z_2$ is detected on $y_4$ if

$$\begin{aligned} \frac{\partial y_4}{\partial z_2} &= y \oplus F(X_4, z_{21} \oplus 1, z_{31} \oplus \frac{\partial z_3}{\partial z_2}dz_2) \\ &= y \oplus F(X_4, \overline{z_{21}}, z_{31} \oplus \frac{\partial z_3}{\partial z_2}dz_2) = 1 \end{aligned} \tag{4}$$

The formula 3 can be used for calculating the influence of the fault at the common fan-out stem $x$ on the output $y$ of the converging fan-out region by consecutive calculating of Boolean derivatives over related FFR chains starting from $x$ up to $y$. For that purpose, for each converging fan-out stem, the corresponding formulas like (3) should be constructed for each converging FFRs. All these formulas will constitute partially ordered computation model for fault simulation. Since the formulas are Boolean, all computations can be carried out in parallel for a bunch of test patterns.

Introduce first the following notations for the formulas above which are used for calculating the Boolean derivatives:

- $(x, y)$ - for $\partial y/\partial x$
- $\{X_k, y\}$ - for a subset of formulas $\{\partial y/\partial x \mid x \in X_k\}$
- $R_{xy}((x, x_1), (x, x_k))$ - for the general case (3)
- $D_x$ - vector which shows if the fault at the node x is detected or not detected at any circuit output
- $DX$ - a set of vectors $D_x$ for the nodes $x \in X$

An example of a computational model of fault simulation for the circuit in Fig.1 is presented in Table I.

The formulas in Table I can be easily created and stored by the topological tracing of the circuit by algorithms developed in [11]. The algorithm has linear complexity. However, the complexity of the computational model and the related fault simulation speed depends on the structure of the circuit. As shown in the papers [14][12], the speed of fault simulation by the proposed parallel critical path tracing method outperforms the speed of the fault simulators of major CAD vendors.

## III. Fault simulation with SSBDDs

The high speed of processing the formulas is achieved by using Structurally Synthesized BDDs (SSBDD) for modeling FFRs [15], [16]. Each FFR $y = F(X)$ is represented by an SSBDD $G$, and each signal path in the FFR represented by a variable $x \in X$ is modelled by a corresponding node in the $G$. All the faults on a signal path collapsed into the faults on the inputs of the FFR, are modelled by the faults at the nodes in $G$. Hence, the targets of the fault simulation are the faults at the SSBDD nodes.

Consider a circuit in Fig. 2, and its corresponding SSBDD. The circuit contains nine signal paths, and each of them is represented by a node in the graph. Note, only the branches of the fan-out inputs are represented in the SSBDD as the model of the FFR. Fault simulation is carried out by traversing the nodes in the graph according to the given test patterns as in the case of traditional BDDs [17].

For simplification the graphical representation of SSBDDs, we use here the following convention: from a node labelled by a variable $x$, the right-hand edge corresponds to the value $x = 1$, and the down-hand edge corresponds to the value $x = 0$. Correspondingly, the exit from the graph to the right means entering the terminal node with constant #1, and the exit from the graph downwards means entering the terminal node with constant #0.

Consider a test pattern 1011101 (1234567) at the inputs of the FFR in Fig. 2. The pattern detects the fault at the input 3 by propagating the faulty signal from the input 3 to the output 8. On the SSBDD in Fig. 2 the edges activated by this pattern are highlighted in bold. The nodes traversed in the graph during simulation of the pattern are marked by gray color. The value on the output 8 of the circuit at this pattern is $y = 1$. Since the nodes 1, $2_2$, 3, 4, $5_2$ are traversed, all they are responsible for the value $y = 1$s, and hence, should be taken as fault candidates in case if the error will be noticed at the circuit output. All other nodes $2_1$, $5_1$, 6, and 7 have not contributed in fault

TABLE I. Levelized fault model equations.

| L | Partially ordered formulas | Types of simulation tasks |
|---|---|---|
| 7 | $\forall x_{4,i} \in X_4 : Dx_{4,i} = \{x_{4,i}, y_4\}$, $Dz_{21} = (z_{21}, y_4), Dz_{31} = (z_{31}, y_4)$; $\forall x_{5,i} \in X_5 : Dx_{5,i} = \{x_{5,i}, y_5\}$, $Dz_{13} = (z_{13}, y_5), Dz_{32} = (z_{32}, y_5)$ | Fault simulation inside the FFRs (F4 and F5) |
| 6 | $Dz_3 = Dz_{31} \vee Dz_{32}$ | Fault simulation of fan-out stems (z3) |
| 5 | $\forall x_{3,i} \in X_3 : Dx_{3,i} = x_{3,i}, z_3 \wedge Dz_3$, $Dz_{22} = (z_{22}, z_3) \wedge Dz_3$, $Dz_{12} = (z_{12}, z_3) \wedge Dz_3$ | Fault simulation inside the FFRs (F3) |
| 4 | $Dz_2 = Rz_2, y_4((z_2, z_{21}) \equiv 1, (z_2, z_{31})) \vee$ $((z_{22}, z_{32}) \wedge Dz_{32})$ | Fault simulation of fan-out stems (z2) |
| 3 | $\forall x_{2,i} \in X_2 : Dx_{2,i} = x_2, i, z_2 \wedge Dz_2$, $Dz_{11} = z_{11}, z_2 \wedge Dz_2$ | Fault simulation inside the FFRs (F2) |
| 2 | $Dz_1 = ((z_1, z_3) \wedge Dz_{31}) \vee$ $Rz_1, y_5((z_1, z_3), (z_1, z_{13}) \equiv 1)$ where $(z_1, z_3) = Rz_1, z_3((z_1, z_{22}), (z_1, z_12) \equiv 1)$ | Fault simulation of fan-out stems (z1) |
| 1 | $\forall x_{1,i} \in X_1 : Dx_{1,i} = x_1, i, z_1 \wedge Dz_1$ | Fault simulation inside the FFRs (F1) |

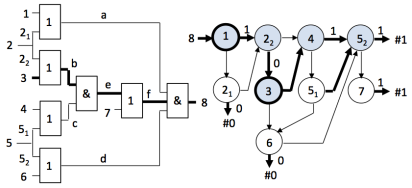


Fig. 2. An FFR of a combinational circuit and its SSBDD.

simulation, and hence, can be excluded from the fault candidates set. Next, by simulating the faults at candidate nodes we can easily notice that only the faults at the nodes 1 and 3 are detected by the given pattern, because at these faults on the graph the terminal node #0 will be reached which means y = 0.

In [11], the algorithms for parallel logic simulation and parallel fault simulation on SSBDDs were proposed. The algorithms are based on the ordering of nodes $m$ by assigning them numerical labels, so that for each node $m$ with label $n(m)$, all its predecessors mj must have labels $n(m_j)$ less than $n(m)$. Logic simulation is based on recursive calculating of the value of the formula

$$D(m) = (x(m) \wedge D(m^1)) \vee (\neg x(m) \wedge D(m^0)), \tag{5}$$

where $D(m)$ for the terminal nodes is equal to the respective constants #1 and #0. Here $x(m)$ denotes the node variable, $m_1$ and $m_0$ are the neighbors of $m$ in directions of $x(m) = 1$, and $x(m) = 0$, respectively. Fault simulation is based on recursive calculating of values of the formulas

$$L(m^1) = L(m^1) \vee (L(m) \wedge x(m)), \tag{6}$$

$$L(m^0) = L(m^1) \vee (L(m) \wedge \neg x(m)), \tag{7}$$

$$S(x(m)) = \frac{\partial y}{\partial x(m)} = L(m) \wedge (D(m^0) \oplus D(m^1)) \tag{8}$$

where $S(x(m)) = 1$ means that the fault at x(m) is detected by the simulated test pattern, otherwise, if $S(x(m)) = 0$, the fault is not detected. Since all the presented formulas are Boolean, the algorithms can be applied by tracing the nodes of the SSBDDs can be applied in parallel for many test patterns, each of them represented by one bit of the computer word. The cost of simulation can be calculated by the number of operations needed for each node of SSBDD. For example, the cost of logic simulation is four operations per node, and the cost of fault simulation is seven operations per node. Hence, to fault simulate the SSBDD in Fig.2 which includes nine nodes, we need 9 * 7 = 63 operations. Example of using the algorithms can be found in [11]. Using SSBDDs instead of the gate-level circuit allows to increase both, the simulation speed for calculating the values of signals in the network of FFRs, and the fault reasoning, since only the collapsed fault set represented by nodes of SSBDDs is processed. This explains the efficiency of the method demonstrated in [11], [12].

## IV. Mixed level fault simulation with SSBDDs

Recently Shared SSBDDs ($S^3$BDD) as a new type of BDDs were proposed to speed-up logic simulation in digital circuits [18], [19]. In the following we propose a two level implementation of the proposed method of critical path tracing, where as the objectives of higher level, the fan-out nodes of the network of FFRs are considered, and as the objectives of lower level, the fan-out branches and fan-out free primary inputs of the network of FFRs are considered. The processing of formulas (3) for calculation of detectability of faults at fan-out nodes is carried out on the higher level using SSBDDs as in Fig. 2, and for computing the detectability of faults at the inputs of FFRs, we will use the data calculated by gate-level logic simulation. To speed up computing of detectability of faults at the inputs of FFRs, we propose to use $S^3$BDDs which can be processed in a similar way as SSBDDs. In Fig. 3, an $S^3$BDD is presented for calculation of the detectability of the faults at the inputs of FFRs. Each entry $x$ in $S^3$BDD corresponds to a node variable $x(m)$ in the SSBDD in Fig.2, and the path from the particular entry to the terminal node represents an AND-function of conditions needed for detectability of the input variable $x$ of the given FFR. For example, the path in Fig.3 from the entry 3 through the nodes $\neg 2_2$, $c$, $\neg 7$, $a$ and $d$ to the terminal node #1 corresponds to the detectability condition of detecting the faults at the $input3$ of the FFR in Fig.2.

The set of these detectability AND-functions for all of the input variables of the given FFR can be easily created from the gate-level structure of the FFR. To combine them in a form of $S^3$BDD like in Fig.3 we can use the algorithm of optimized $S^3$BDD synthesis developed in [19].

The cost of fault simulation using $S^3$BDDs can be calculated in terms of the number of operations needed, and is equal to $C = N - N_T$ where $N$ is the number of all nodes in the $S^3$BDD model, and $N_T$ is the number of end nodes of the model. For the $S^3$BDD model in Fig.3 we have $C = 17 - 2 = 15$, which is four times less than 63 operations needed for simulation of the SSBDD in Fig. 2. Consider, as an example, the mixed level work share in the computing processes of the level 2 in Table I between SSBDD and $S^3$BDD models. These processes handle the critical path tracing over the nested configuration of three fan-out re-convergence areas. In the process

$$(z_1, z_3) = R_{z_1}, z_3((z_1, z_{22}), (z_1, z_{12})1), \tag{9}$$

$(z_1, z_{22})$ is computed at the low-level on the $S^3$BDD for the FFR with output $z_2$, whereas $R_{z_1}$,$z_3$ is computed at the higher level using the SSBDD of $z_3$ after the following updates of the node variable values: $z_2 2 = z_2 2 \oplus (z_1, z_2 2)$, and $z_1 2 = \neg z_1 2$. On the other hand, in the process

$$D_{z_1} = ((z_1, z_3) \wedge D_{z_{31}}) \vee R_{z_1}, y5((z_1, z_3), (z_1, z_{13})1), \tag{10}$$

$D_{z_{31}}$ is computed at the low-level on the $S^3$BDD for the FFR with output $y_4$, whereas $R_{z_1}$,$y_5$ is computed at the higher level using the SSBDD of $y_5$ after the following updates: $z_{32} = z_{32} \oplus (z_1, z_3)$, and $z_{13} = \neg z_1$.

Additional side-effect of the mixed-level fault reasoning is the increase of the accuracy in reporting the detected faults. Using the information about the gate-level structure of FFRs, allows to specify the detected faults inside the FFRs. For example, the entries $a'$ and $d'$ in the $S^3$BDD in Fig.3 are introduced to mark the sub-graphs for calculating the detectability of internal gate-level faults at the nodes $a$ and $d$, respectively, inside the FFR, presented in Fig.2. Similar entries

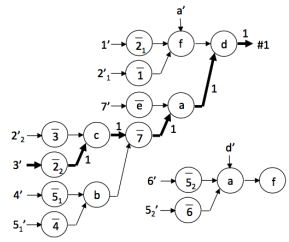


Fig. 3. Direct fault simulation using $S^3$BDDs.

can be added in Fig.3 for other internal nodes $b$, $c$, $e$, and $f$ in the same FFR.

The speed-up in mixed-level fault reasoning and the increasing accuracy of detected fault reporting is accompanied with additional time cost needed for logic simulation of FFRs at the gate-level. However, when comparing the total times for logic simulation and fault simulation this payload increase will be negligible.

## V. Reordering the Computing Model using Levels

As was mentioned earlier circuit partitioning technique into levels for concurrent execution have been already used before [20][22][23]. The level $i$ gate is defined in [20] as one having primary inputs of the circuit and outputs of level $k$ gates as its inputs, such that $k < i$. However in [22], which cites the previous paper the definition is slightly different, stating that level of a gate represents its distance in gates from primary inputs (PI's) of the circuit. This definition is more strict in the sense that one of the inputs of the level $i$ gate, must originate from the level $i-1$, if $i \neq 0$. This difference however is crutial for parallelisation, because the use of the first definition could potentially result in bigger number of levels with fewer gates in them. As levels should be evaluated sequentially - this could decrease the amount of parallelism dramatically. In our case, as we deal with FFRs, we would stick to the second definition and rephrase it for our purpose.

Both, the logic simulation model and the computational model for fault back-tracing described in Section II are presented as networks of partially ordered formulas linked to each other by variables and computed using SSBDDs. Here and throughout the paper we would use the word *variable* to indicate these elements of the circuit. The level of variable is its distance in variables from PI's or, in other words, the level i variable should have at least one of its inputs originating from level i-1 variable, if $i \neq 0$.

In the computational model, the variables are numbered in serial fashion starting at primary inputs and finishing at primary outputs. Variables are serialized such that each input of variable i is the output of variable k, where $k < i$. This is very similar to the first definition of levels from [20].We are using OpenCL framework for parallel execution[21]. Therefore it is necessary to define regions of variables, belonging to the same level, as sub-array. Only variables of particular level must be included into sub-array. If variable $x$ belongs to level $i$, then level $i$ should be represented as a continuous sequence of variables starting from variable $x$ to variable $y$, such that every variable $z$ ($x \leq z < y$) belongs to level $i$ and variable $y$ belongs to level $i+1$. This is why it is necessary, to reorder the variables according to our definition of levels. Note that this operation is only required once and does not belong to fault simulation process. The reordered computational model can be saved as a file and used later for simulation, without a need to repeat this step.

Fault model represents segments of critical path to be simulated. Each segment starts at the output of particular variable and ends at primary output of the circuit. Therefore there is one-to-one correspondence between critical path segments to be fault simulated and particular variable. This fact makes it possible to use levelized structure of computational model for Fault model as well. It is important because using levels we could analyze critical path segments starting at the same level in parallel, thus speeding up the fault simulation.

OpenCL framework requires single program for all the parallel devices, which would manipulate on different data. Such program is called kernel. It is executed on all available devices in parallel for all variables inside a single level. The best way to provide the data for kernel is an array. During preparation of the computational model the variable indexes are placed into an array according to their levels. The kernel only requires to know the offset of the level inside the array of variable indexes and the size of this level. Host CPU schedules the kernel executions level by level into the OpenCL execution queue. The execution in the queue is strictly ordered, so that OpenCL driver handles the synchronisation between consecutive kernel executions. This ensures that all variables of the current level have been computed, before moving to the next level.

## VI. Results

The experiments were carried out on IBM System x3500 M3 7380 Server (2x 6-core Xeon E5690 running at 3,47Ghz with hyperthreading) using 64-bit Novell SuSe Linux Enterprise Server 11 x86_64. This system has 12 physical CPU cores, 12 virtual hyperthreading cores and 96Gb of RAM. Simulation times were calulated for the sets of 10000 random test patterns. The circuits from three benchmark suites ISCAS'85, ISCAS'89, ITC'99 were simulated. The same circuits as in [12] were chosen in order to compare the results.

Concurrent execution time $T_p$ of PECPT fault simulation can be divided into two parts: $T_p = T_o + T_c$. The first part is the time $T_o$, which we would call *concurrency overhead*. This is required to make a transition from "single thread"- to "multiple thread"-execution and back again. This time slot involves creation of multiple threads, allocating additional memory, synchronisation at the end of computation and transition back to single thread. The second part is time $T_c$, which is pure *computation time* required by all threads to deliver a result. This time can be seen in Table II and can be treated as a lower possible bound for concurrent computation. The concurrency overhead $T_o$ depends on the amount of parallel hardware used and increases with number of CPUs. The computation time $T_c$ depends on the amount of computation required. Parallel simulation time $T'_p = T_p + T_{fm} + T_{ff}$, where $T_{fm}$ is the time required to compile the fault model and $T_{ff}$ is the time of fault-free logic simulation.

Table II shows the results of the PECPT execution time $T'_p$ in comparison to PPECPT $T_{PPECPT}$ [14]. As we see, the new method outperforms considerably the previous method, and the gain increases with the size of the circuit (up to the order of magnitude in case of the circuit b19 containing 450 thousands gates). Amount of calculation for small circuits is small, which makes overall execution time $T_p$ large in comparison to computation time $T_c$. This can be expressed by overhead ratio $R = T_p/T_c$ and is clearly seen from results in Table II. Overhead ratio $R$ for the case of maximum acceleration is also brought in the table to see the concurrency overhead for different circuits. It can be seen from the Table II that overhead ratio is getting closer to one, with growth of the circuit size. In case of circuit b19 the speedup gets almost identical to ideal, because $T_p$ and $T_c$ become almost equal.

Along with execution time there are two speedup values we compute for every benchmark. These are $S_p$ and $S_c$. Both include single CPU (non-parallel) computation time of fault model $T_{tpl}$ and fault-free simulation $T_{ffs}$ of the circuit. Along with these $S_p$ uses parallel execution time $T_p$ for its computation and $S_c$ uses pure parallel computation time $T_c$. The equations for speedup values $S_p$ and $S_c$ are as following:

$$S_p = \frac{T_{PPECPT}}{T_{tpl} + T_{ffs} + T_p} = \frac{T_{PPECPT}}{T'_p}$$

$$S_c = \frac{T_{PPECPT}}{T_{tpl} + T_{ffs} + T_c} = \frac{T_{PPECPT}}{T'_c}$$

$S_c$ can be though as topmost ideal case of speedup by PECPT algorithm. It can be seen from the results that smaller circuits achieve

TABLE II. EXECUTION TIMES OF PPECPT AND PECPT.

| Circuit | $T_{ppecpt}$, s | Concurrency overhead (PECPT) | | | | Pure computation (PECPT) | | |
|---|---|---|---|---|---|---|---|---|
| | | $T'_p$, s | $S_p$ | $S_p$ #cpu | R | $T'_c$ s | $S_c$ | $S_c$ #cpu |
| c1908 | 0,0568 | 0,0846 | 0,67 | 6 | 2,86 | 0,0330 | 1,72 | 5 |
| c2670 | 0,0405 | 0,0873 | 0,46 | 4 | 6,52 | 0,0334 | 1,21 | 6 |
| c3540 | 0,1830 | 0,1315 | 1,39 | 8 | 1,81 | 0,0754 | 2,43 | 7 |
| c5315 | 0,0849 | 0,0922 | 0,92 | 4 | 3,05 | 0,0487 | 1,74 | 5 |
| c6288 | 1,4610 | 0,6211 | 2,35 | 6 | 1,61 | 0,3883 | 3,76 | 8 |
| c7552 | 0,1545 | 0,1187 | 1,30 | 6 | 1,94 | 0,0718 | 2,15 | 6 |
| s13207 | 0,1798 | 0,1332 | 1,35 | 5 | 5,05 | 0,0857 | 2,10 | 10 |
| s15850 | 0,4714 | 0,2107 | 2,24 | 8 | 2,34 | 0,1370 | 3,44 | 7 |
| s35932 | 0,2554 | 0,1739 | 1,47 | 10 | 1,95 | 0,1381 | 1,85 | 12 |
| s38417 | 0,7453 | 0,2427 | 3,07 | 12 | 1,95 | 0,1869 | 3,99 | 12 |
| s38584 | 0,5945 | 0,2492 | 2,39 | 9 | 2,43 | 0,1791 | 3,32 | 12 |
| b14 | 2,7742 | 0,8752 | 3,17 | 8 | 1,29 | 0,7300 | 3,80 | 9 |
| b15 | 5,0420 | 1,1771 | 4,28 | 10 | 1,49 | 0,9258 | 5,45 | 10 |
| b17 | 14,8550 | 2,4053 | 6,18 | 20 | 1,29 | 2,1121 | 7,03 | 12 |
| b18 | 67,3279 | 7,1499 | 9,42 | 24 | 1,09 | 6,7738 | 9,94 | 24 |
| b19 | 147,6501 | 14,4685, | 10,20 | 24 | 1,03 | 14,0707 | 10,49 | 24 |

small or negative speedup. On the other hand bigger circuits take advantage of higher number of processors. Such poor result for smaller circuits can be explained by low amount of parallelism accompanied by high overhead ratio R. Both of the factors change positively when circuit size becomes bigger. One of the challenges of this method is that different number of processors is required to achieve maximum speedup for different circuits. The number of processors used to achieve maximum speedup is brought under #cpu columns. It can be seen that this number grows along with the circuit size.

Speedup $S_p$ dependence on the number of processors is shown in Fig.4a (ISCAS'85), Fig.4b (ISCAS'89), Fig.4c (ITC'99). The fluctuation in speedup of some circuits can be explained by the fact, that it is up to OpenCL runtime to decide which processors to use for execution. Because our testsystem has virtual hyperthreading cores they can also be arbitrarily chosen for execution, which could influence the speed of execution in situations where less physical cores are used for computation, although the overall number of cores is bigger. For all the benchmarks we can see that after the limit of physical cores is reached the speedup is starting to decline or stays the same. On the ITC'99 benchmarks b18 and b19 it is slightly increasing, when more than 12 cores are used. This shows that the larger the circuits, the more number of cores can be exploited to achieve the maximum speed-up of simulation.

We compared PECPT to single processor simulators, such as FSIM, PPECPT and commercial simulators C1 and C2. We normalised execution time of all the simulators using previous results from [12] and execution time of PPECPT from Table II, because PECPT was executed on different hardware. The comparison is shown in Table III.

PECPT proves to be in average 3.8 times quicker than FSIM and around two times - than PPECPT for relatively smaller ISCAS benchmarks. The speedup over commercially available simulators is more than eight times over C1 and two orders of magnitude over C2. When ITC'99 benchmark circuits are also taken into consideration the average speedup over PPECPT grows to 4.0 and over C1 even 8.7 times in average, which suggests that simulation of bigger circuits benefits more from our method.

We have also compared PECPT speedup results to GPU based parallel fault simulator and fault table generater GFTABLE [24]. GFTABLE is pattern parallel simulator, which uses bit- and thread-level PP to boost the performance of uniprocessor simulator FSIM. We have used the results from Table 4 in [12] to normalize PECPT speedup. Normalization is required because PECPT speedup is computed in relation to PPECPT, while GFTABLE speedup is computed in relation to FSIM. As there is no FSIM execution time provided for ITC'99 benchmarks, we have taken the average ratio of 1.7 reported in [12] to normalize PECPT results for those circuits.

The Fig. 5 shows speedup in comparison to uniprocessor version of FSIM for both algorithms. We have arranged circuits in sequence where their corresponding number of gates is growing. This way we could clearly see the speedup dependency on the size of the circuit. It can be seen that PECPT proves to be more beneficial on the circuit sizes comparable to ITC'99 benchmarks. For example for circuit c5315 from ISCAS'85 the speedup is 8.03 for GFTABLE and 1.50 - PECPT, while for the ITC'99 circuit b15 the speedup is 2.57 for GFTABLE and 7.28 - PECPT. It is interesting to note that results of the GFTABLE decrease when circuit size is growing, while PECPT in opposite gives less gain in speed-up, while circuit size is smaller.

TABLE III. EXECUTION TIME COMPARISON.

| circuit | #fanouts | #branches | | Simulation time,s | | | | |
|---|---|---|---|---|---|---|---|---|
| | | max | avg | fsim | c1 | c2 | ppecpt | pecpt |
| c2670 | 290 | 28 | 3,7 | 0,081 | 0,223 | 2,430 | 0,041 | 0,087 |
| c3540 | 356 | 22 | 4,5 | 0,407 | 1,505 | 8,745 | 0,183 | 0,132 |
| c5315 | 510 | 31 | 5 | 0,149 | 0,594 | 6,047 | 0,085 | 0,092 |
| c6288 | 1456 | 16 | 2,6 | 2,389 | 5,489 | 56,072 | 1,461 | 0,621 |
| c7552 | 812 | 72 | 4,1 | 0,348 | 1,043 | 11,332 | 0,155 | 0,119 |
| s13207 | 1224 | 37 | 3,7 | 0,225 | 0,503 | 6,291 | 0,180 | 0,133 |
| s15850 | 1518 | 34 | 3,6 | 0,943 | 2,112 | 19,379 | 0,471 | 0,211 |
| s35932 | 5295 | 1449 | 3,4 | 0,412 | 1,058 | 17,477 | 0,255 | 0,174 |
| s38417 | 4569 | 49 | 3,2 | 1,725 | 3,343 | 33,007 | 0,745 | 0,243 |
| s38584 | 3946 | 88 | 4,5 | 1,124 | 2,155 | 29,727 | 0,595 | 0,249 |
| *Average speedup* | | | | *3,786* | *8,748* | *92,460* | *2,024* | *1,000* |
| b14 | 2409 | 82 | 4,8 | n/a | 9,413 | n/a | 2,774 | 0,875 |
| b15 | 2353 | 95 | 4,8 | n/a | 7,411 | n/a | 5,042 | 1,177 |
| b17 | 8145 | 149 | 4,8 | n/a | 22,340 | n/a | 14,855 | 2,405 |
| *Average speedup* | | | | *n/a* | *8,774* | *n/a* | *4,118* | *1,000* |

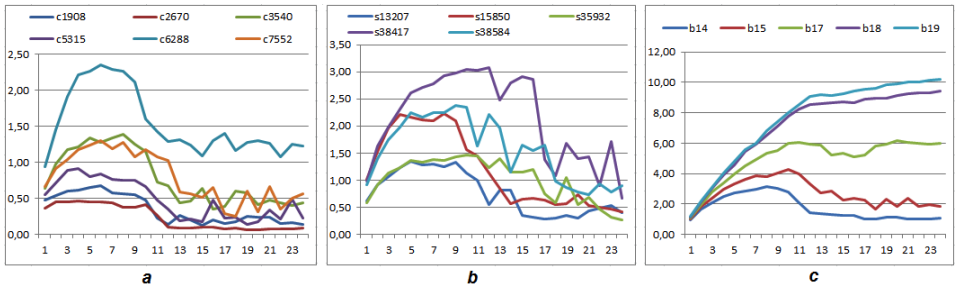
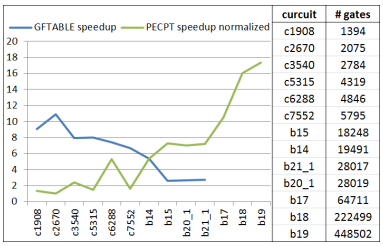



Fig. 4. Speedup vs #CPU for PECPT. a). ISCAS'85 benchmarks, b). ISCAS'89 benchmarks, c). ITC'99.




| curcuit | # gates |
|---|---|
| c1908 | 1394 |
| c2670 | 2075 |
| c3540 | 2784 |
| c5315 | 4319 |
| c6288 | 4846 |
| c7552 | 5795 |
| b15 | 18248 |
| b14 | 19491 |
| b21_1 | 28017 |
| b20_1 | 28019 |
| b17 | 64711 |
| b18 | 222499 |
| b19 | 448502 |

Fig. 5. Comparison of GFTABLE and PECPT.

It is stated in [24], that performance of GFTABLE for bigger circuits is influenced by amount of global memory available on GPU. This highlights the scalability bottelneck of the GFTABLE. The results of our approach also depend on the amount of system memory available, but CPU systems in general are more flexible in increasing memory size than GPUs. Even for the circuits, which could fit into GPU memory we can see slight decrease in performance of GFTABLE. Contrary the results of our approach in average become better while circuit size increases.

### A. Future work

In order to make the method more practical it is needed to define the number of CPUs involved to provide the best speedup for particular circuit. We believe this can be achieved by further research because the optimum number of CPUs and speedup depend on circuit parameters.

## VII. Conclusion

We have proposed a new method for concurrent pattern parallel exact critical backtracing based fault simulation by exploiting circuit processing concurrency. The first time, the parallelization in fault simulation is carried out simultaneously in three dimensions: pattern parallelism, fault parallelism and computing model parallelism, where the pattern- and fault-parallelism are utilized using each single CPU core, while computing model parallelism is achieved using multiple CPUs.

A novel mixed level technique for fault reasoning was proposed to speed up and to increase the accuracy of fault simulation, compared with previous methods.

Experiments showed that the average speed-up compared to the best uniprocessor based simulators is around 3-4 times in average, and up to order of magnitude compared to the available state-of-the-art commercial uniprocessor based simulators. The method is well scaling, the speed up of the method grows with the size of the circuit, opposite to the pattern-parallel simulation method. The reason lies in the memory bottleneck of shared-memory systems, which increases more rapidly for pattern-parallel systems with the growth of the circuit size.

## Acknowledgment

The work was supported in part by EU FP7 STREP project BASTION, Estonian ICT project FUSETEST, by EU through the European Structural and Regional Development Funds, by the Estonian Doctoral School in Information and Communication Technology and by the IT Academy Program of Information Technology Foundation for Education.

## References

[1] J. A. Waicukauski and et al.,*Fault simulation for structured VLSI*, VLSI Systems Design, pp.20-32, Dec. 1985

[2] B. Underwood, J. Ferguson. *The Parallel Test Detect Fault Simulation Algorithm*. ITC, pp.712-717, 1989

[3] M. Abramovici, P. R. Menon, D. T. Miller. *Critical Path Tracing An Alternative to Fault Simulation*. Proc. 20th Design Automation Conference, pp. 2-5, 1987.

[4] K. J. Antreich, M. H. Schulz. *Accelerated Fault Simulation and Fault Grading in Combinational Circuits*. IEEE Trans. On Computer-Aided Design, Vol. 6, No. 5, pp.704-712, 1987.

[5] F. Maamari, J. Rajski. *A Method of Fault Simulation Based on Stem Region*. IEEE Trans. on CAD, Vol.9, No.2, pp. 212-220, 1990.

[6] D. Harel, R. Sheng, J. Udell. *Efficient Single Fault Propagation in Combinational Circuits*. Int. Conf. on CAD, pp.2-5, 1987.

[7] D. B. Armstrong. *A deductive method for simulating faults in logic circuits*. IEEE Trans. Comp., C-21(5), 464-471, 1972.

[8] E. G. Ulrich, T. Baker. *Concurrent simulator of nearly identical digital networks*. IEEE Trans.on Comp.,7(4), pp.39-44, 1974.

[9] W. T. Cheng, M. L. Yu. *Differential fault simulation: a fast method using minimal memory*. DAC, pp.424-428, 1989.

[10] L. Wu,D. M.H.Walker, *A Fast Algorithm for Critical Path Tracing in VLSI Digital Circuits*, 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems(DFT'05), 3-5 October, 2005,pp.178-186.

[11] R. Ubar, S. Devadze, J. Raik, A. Jutman. *Ultra Fast Parallel Fault Analysis on Structural BDDs*. 12th IEEE European Test Symposium ETS 2007, Freiburg, Germany, May 20-24, 2007.

[12] R. Ubar, S. Devadze, J. Raik, A. Jutman. *Parallel X-Fault Simulation with Critical Path Tracing Technique*. IEEE Conf. Design, Automation & Test in Europe - DATE-2010, Dresden, Germany, March 8-12, 2010, pp. 1-6.

[13] Thayse, *Boolean Calculus of Differences*, Springer Verlag, 1981.

[14] R. Ubar, S. Devadze, J. Raik, A. Jutman. *Parallel Fault Backtracing for Calculation of fault Coverage*. 13th Asia and South Pacific Design Automation Conference - ASP-DAC 2008, Seoul, Korea, Jan. 21-24, 2008, pp. 667-672.

[15] R. Ubar. *Test Synthesis with Alternative Graphs*. IEEE Design and Test of Computers. Spring, 1996, pp.48-59.

[16] R. Ubar. *Combining Functional and Structural Approaches in Test Generation for Digital Systems*. Journal of Microelectronics and Reliability, Elsevier Science Ltd. Vol. 38:3, pp.317-329, 1998.

[17] Minato, S. *BDDs and Applications for VLSI CAD*. Kluwer Academic Publishers. 1996.

[18] D. Mironov, R. Ubar, J. Raik. *Logic Simulation and Fault Collapsing with Shared Structurally Synthesized BDDs*. IEEE European Test Symposium, Paderborn, Germany, May 26-30, 2014.

[19] D. Mironov, R. Ubar. *Lower Bounds of the Size of Shared Structurally Synthesized BDDs*. IEEE 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS). Warsaw, April, 23-25, 2014, pp. 77-82.

[20] Amin, M.B. ; Vinnakota, B., *Data parallel fault simulation*. International Conference on Computer Design: VLSI in Computers and Processors (ICCD '95), 1995, pp. 610-615.

[21] *OpenCL standard for parallel programming of heterogenious systems.* https://www.khronos.org/opencl/

[22] Varshney, A.K., Vinnakota, B., Skuldt, E., Keller, B., *High Performance Parallel Fault Simulation*. International Conference on Computer Design 2001 (ICCD 2001), 2001, pp. 308-313.

[23] Gulati, K., Khatri, S.P., *Towards Acceleration of Fault Simulation using Graphics Processing Units*. 45th ACM/IEEE Design Automation Conference 2008 (DAC 2008), 2008, pp. 822-827.

[24] Gulati, K. ; Khatri, S.P., *Fault Table Generation using Graphics Processing Units*. IEEE International High Level Design Validation and Test Workshop 2009 (HLDVT 2009), 2009, pp. 60-67.

# APPENDIX E

# Combinational Fault Simulation in Sequential Circuits

Raimund Ubar, Jaak Kõusaar, Maksim Gorev, Sergei Devadze
*Department of Computer Engineering, TTU, Ehitajate tee 5, 19086 Tallinn, Estonia*
*E-mails: raiub@pld.ttu.ee, jaak.kousaar@gmail.com, maksim.gorev@ttu.ee, serega@pld.ttu.ee*

*Abstract.* **We propose a very fast fault simulation method which is based on exact parallel critical path tracing developed for combinational circuits. To convert the sequential problem of fault simulation into the combinational one we introduce into the circuit a set of MISRs to improve the circuit's observability. The role of these MISRs is to monitor signals on the global feedback loops, and on selected fan-out stems in the circuit. The given sequential circuit is partitioned into a set of sequential or combinational sub-circuits, with breakpoints at global feedback loops or at selected fan-out stems. The simulated test sequence is mapped into local sets of input patterns applied to the sub-circuits. For these local test patterns, each sub-circuit is fault simulated by exact parallel critical path tracing similarly as a combinational equivalent circuit. The feasibility and correctness of the method is shown, and the experimental results which demonstrate the speed-up achieved by the method are provided.**

*Keywords:* sequential circuits, stuck-at-faults, design for testability, fault simulation with critical path tracing

## I. Introduction

Fault simulation is one of the most important tasks in digital circuit design and test. The efficiency of test quality and dependability evaluation, test generation and fault diagnosis relies heavily on the speed of fault simulation. Accelerating fault simulation would have a strong impact to all of the mentioned applications.

Many different methods have been proposed for fault simulation in combinational circuits based on the concept of parallel pattern single fault propagation (PPSFP) [1]. Another trend is based on the fault reasoning (deductive [2], concurrent [3] and differential simulation [4]) used to be very powerful, since these methods allow to collect all detectable faults by a single run of the given test pattern. What they cannot do, is to produce reasoning for many test patterns in parallel.

The original critical path tracing method [5, 6] eliminates explicit fault simulation for faults within Fan-out-Free Regions (FFR). However, the explicit simulation of faults at fan-outs was still needed. A modified critical path tracing technique that excludes fault simulation for fan-out stems, and includes a system of rules to check the exactness of critical path tracing beyond the FFRs, and which is linear in time, was proposed in [7]. However, the rule based strategy does not allow parallel analysis and rule check for many patterns simultaneously. This drawback was removed in [8] by introducing a novel concept of Parallel Pattern Exact Critical Path Tracing (PPECPT) which can be applied efficiently also beyond FFRs. In [9], the same method was for a general class of X-faults. The main idea of the method was in compiling a dedicated compact computing model through the circuit topology analysis, which allows exact critical path tracing throughout the full circuit and not only inside FFRs.

Unfortunately, for sequential circuits the parallelism in fault simulation and fault reasoning is not possible, because of the sequential (time related) dependence of signals in the circuit. In this paper we propose to modify the given circuit to improve the transparency (observability) of the circuit. The traditional way to do that is to use the scan-path concept [10] which converts the sequential problem of fault simulation to the combinational one. However, the use of scan-chains has proven to be often inadequate due to increasing the cost in terms of additional hardware and increased testing time [11], excessive power dissipation during test [12] and leading to yield loss because of over-testing [13].

In the following we show that a sequential circuit can still be fault simulated as a combinational one when to improve its observability by inserting a set of Multiple Input Signature Registers (MISR), for monitoring of a selected subset of test points in the circuit. We introduce and discuss two rules for selecting these test points for including MISRs, and then show how the test sequence can be mapped into a set of independent local test sequences which can be simulated in parallel similarly to the case of combinational circuits.

The target of the paper is to combine three ideas: to suggest functional testing of sequential circuits to be carried out at-speed and on-line, instead of scan-path testing, for providing better test quality; to improve observability (testability) of the circuit with better fault diagnostic resolution; and, finally, to provide a method of fault simulation in a modified circuit with a dramatic speed-up compared to the traditional non-parallel fault simulation of sequential circuits.

We consider in this paper only the class of stuck-at-faults (SAF), however, as shown in [9], the results can be extended to other fault classes like conditional SAF, transition delays, and *X*-faults.

The rest of the paper is organized as following. In Section 2 we present the theoretical basics of the topic by giving a short overview of the exact parallel critical path tracing in combinational circuits were we show how the fault tracing can be expanded in exact way beyond the fan-out stems. In Section 3 we describe how this method can be generalized for the case of sequential circuits. In Section 4, we describe experimental results, and Section 5 concludes the paper.

## II: PARALLEL PATTERN CRITICAL PATH TRACING

Consider a combinational circuit as a network of FFRs where each of the FFRs can be represented as a Boolean function $y = F(x_1, x_2, \dots x_n) = F(X)$, where $X = x_1, x_2, \dots x_n$ is the input vector of the FFR. Such a network is presented in Fig.1.

The fault simulation for a FFR according to traditional critical path tracing is equivalent to calculation of Boolean derivatives: if $\partial y/\partial x = 1$ then the fault is propagated from $x$ to $y$. This check can be performed in parallel for a given subset of test patterns. In order to extend the parallel critical path tracing beyond the fan-out free regions we use the concept of Boolean differentials [14].

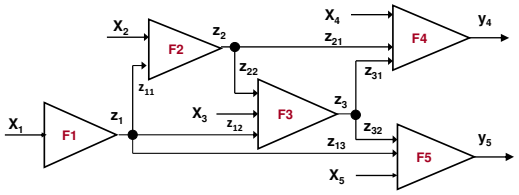



**Fig.1.** Combinational circuit with five FFRs

Consider the full Boolean differential of the FFR given by $y = F(X)$ as

$$dy = y \oplus F((x_1 \oplus dx_1),\dots,(x_n \oplus dx_n)) = y \oplus F(X \oplus dX) \qquad (1)$$

Here, by $dx$ we denote the change of the value of $x$ because of a fault at $x$, whereas $dy = 1$ if some erroneous change of the values of arguments of the function (1) due to a fault causes the change of the value of $y$, otherwise $dy = 0$.

Let $x$ be a fan-out variable with branches which converge in a FFR $y = F(X)$ at the inputs denoted by a subset $X' \subset X$ . In [11] we have shown that from the expression (1) the following relationship can be derived:

$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}),\dots,(x_n \oplus \frac{\partial x_n}{\partial x})) = y \oplus F(X \oplus \frac{\partial X}{\partial x})$$

or taken in the vector form as

$$\frac{\partial y}{\partial x} = y \oplus F(X' \oplus \frac{\partial X'}{\partial x}, X'') \qquad (2)$$

where $X' \subset X$ is the sub-vector of variables which depend on $x$, and $X'' = X \setminus X'$ is the sub-vector of variables which do not depend on $x$.

For example, to get to know if the fault on $z_2$ in the circuit of Fig.1 can be detected on $y_4$ by the given pattern, we have to check if

$$\frac{\partial y_4}{\partial z_2} = y \oplus F(X_4, z_{21} \oplus 1, z_{31} \oplus \frac{\partial z_3}{\partial z_2}) = y \oplus F(X_4, \overline{z_{21}}, z_{31} \oplus \frac{\partial z_3}{\partial z_2}) = 1$$

The formula (2) can be used for calculating the impact of the fault at the fan-out stem $x$ on the output $y$ of the converging fan-out region by consecutive calculating of Boolean derivatives over related FFR chains starting from $x$ up to $y$. For that purpose, for each converging fan-out stem, the corresponding formulas like (3) should be constructed for each FFR involved in the convergence. In the case of nested convergences, the formulas will have as well a nested structure. All these formulas will constitute partially ordered computation model for fault simulation which can be composed by the topological analysis of the circuit [9]. Since the formulas are Boolean, all computations can be carried out in parallel for a bunch of test patterns.

Introduce the following notations for representing symbolically the computing model for fault simulation using the formula (3):

- $(x, y)$ – for $\partial y/\partial x$,
- $\{X_k, y\}$ – for a subset of formulas $\{\partial y/\partial x \mid x \in X_k\}$
- $R_{xy}((x, x_1), \dots (x, x_k))$ – for the general case (3), where $X' = (x_1,\dots,x_k)$,
- $Dx$ – vector which shows if the fault at the node $x$ is detected or not detected at any circuit output,
- $DX$ – a set of vectors $Dx$ for the nodes $x \in X$.

An example of a computational model, using the given symbolics, for the full fault simulation of the circuit in Fig.1, is presented in Table 1.

Table I. Computational model for fault simulation

| L | Partially ordered formulas | Types of simulation tasks |
|---|---|---|
| 7 | $\forall x_{4,i} \in X_4$: $Dx_{4,i} = \{x_{4,i}, y_4\}$, $Dz_{21} = (z_{21}, y_4)$, $Dz_{31} = (z_{31}, y_4)$; $\forall x_{5,i} \in X_5$: $Dx_{5,i} = \{x_{5,i}, y_5\}$, $Dz_{13} = (z_{13}, y_5)$, $Dz_{32} = (z_{32}, y_5)$ | Fault simulation inside the FFRs ($F_4$and $F_5$) |
| 6 | $Dz_3 = Dz_{31} \vee Dz_{32}$ | Fault simulation of fan-out stems ($z_3$) |
| 5 | $\forall x_{3,i} \in X_3$: $Dx_{3,i} = \{x_{3,i}, z_3\} \wedge Dz_3$ $Dz_{22} = (z_{22}, z_3) \wedge Dz_3$, $Dz_{12} = (z_{12}, z_3) \wedge Dz_3$ | Fault simulation inside the FFRs ($F_3$) |
| 4 | $Dz_2 = R_{z2,y4}((z_2, z_{21}) \equiv 1, (z_2, z_{31})) \vee ((z_{22}, z_{32}) \wedge Dz_{32})$ | Fault simulation of fan-out stems ($z_2$) |
| 3 | $\forall x_{2,i} \in X_2$: $Dx_{2,i} = \{x_{2,i}, z_2\} \wedge Dz_2$, $Dz_{11} = \{z_{11}, z_2\} \wedge Dz_2$ | Fault simulation inside the FFRs ($F_2$) |
| 2 | $Dz_1 = ((z_1, z_3) \wedge Dz_{31}) \vee$ $R_{z1,y5}((z_1, z_3), (z_1, z_{13}) \equiv 1)$ where $(z_1, z_3) = R_{z1,z3}((z_1, z_{22}), (z_1, z_{12}) \equiv 1))$ | Fault simulation of fan-out stems ($z_1$) |
| 1 | $\forall x_{1,i} \in X_1$: $Dx_{1,i} = \{x_{1,i}, z_1\} \wedge Dz_1$ | Fault simulation inside the FFRs ($F_1$) |

The formulas presented in Table 1 can be easily created and stored by the topological tracing of the circuit by algorithms developed in [9]. The algorithm has linear complexity. However, the complexity of the computational model and the related fault simulation speed depends on the structure of the circuit.

## III. CONVERTING THE SEQUENTIAL FAULT SIMULATION TASK INTO THE COMBINATIONAL ONE

The substantial problem of fault simulation in sequential circuits lies in the fact that the same fault can influence on a particular component in different time frames. This fact excludes the possibility of exploiting the powerful critical path tracing based method, explained in the previous section, for fault simulation in combinational circuits. The reason is in the exponential explosion of the number of nested and intersected re-converging fan-out regions over different time-frames. However, this problem as we will show can be removed if there will be a possibility to detect the fault in the first occasion when it has propagated up to the component.

There are two reasons why a fault can be propagated to the same component during different time frames: because of the global feedback which includes this component, and because of a re-convergent fan-out where the fault may propagate from the fan-out stem to the converging point by different number of clocks. If we will insert a MISR to these "problem causing" test points, the fault can be captured always at the first occasion it influences on the component. The detection of the fault is fixed, and we can ignore its impact in the future. Note, we consider here only the problem of fault detection (for measuring the fault coverage), and not the task of creating fault tables to be used for fault diagnosis purposes.

From above, two rules result for improving the observability of the sequential circuit:

RULE 1: Insert a MISR to all registers (and only to them) which are included into a global feedback. Inserting a MISR is equivalent to cutting the feedback loop (in a sense to ignore the further fault propagation).

RULE 2: Insert a MISR into all fan-out stems which have at least a single converging point, so that a fault may propagate from the fan-out stem to this point by different number of clocks.

Consider a sequential circuit in Fig.2 which consists of 9 registers (latches) $R_1 - R_9$, and 8 combinational sub-circuits $F_1 - F_9$. The circuit has 5 inputs and 2 outputs.

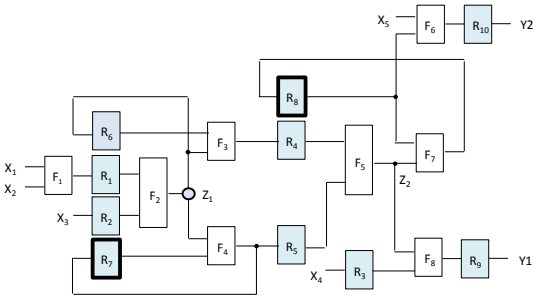



**Fig.2.** Sequential circuit

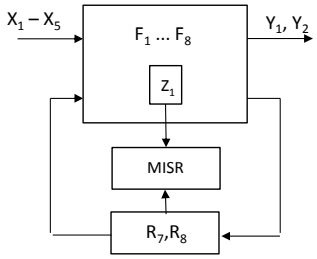



**Fig.3.** Sequential circuit with MISR

In the circuit in Fig.2, two registers $R_7$ and $R_8$ are included into a global feedback loop, and hence, according to RULE 1, they must be furnished by MISR. On the other hand, there is a fan-out stem $Z_1$ which has two branching paths which re-converge in $F_3$. The first path represents a direct connection, and the second one is a path via register $R_6$, where the possible faulty signal needs for propagating from $Z_1$ to $F_3$ additional clock. Hence, according to RULE 2, the node $Z_1$ must be monitored by MISR. The modified circuit is presented in Fig.3. For better focusing to the problem under discussion, and to skip the technical question of handling don't care signals, we assume that the registers with global feedback R7 and R8 are provided with RESET inputs $RES_7$ and $RES_8$, respectively.

In Fig.4, a simulation cycle of a single independent test sequence with lengths of 6 clocks is shown where by rectangles the 5 observation points are denoted. In this simulation cycle we can extract 5 functions (the upper indexes denote the delay in clock cycles between the moments when the values of argument signals and the function signal were fixed, respectively):

$$
\begin{aligned}
Z_1 &= f_{z1}(X_1^{-1}, X_2^{-1}, X_3^{-1}) \\
R_7 &= f_{R7}(RES_7^{-2}, Z_1^{-1}) \\
R_8 &= f_{R8}(RES_8^{-2}, R_7^{-2}, Z_1^{-2}, Z_1^{-3}) \\
Y_1 &= f_{Y1}(R_7^{-2}, Z_1^{-2}, Z_1^{-3}, X_4^{-2}) \\
Y_2 &= f_{Y2}(X_5^{-1}, R_8^{-1})
\end{aligned}
\qquad (3)
$$

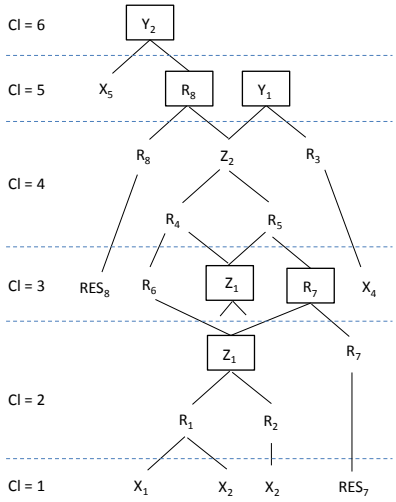



**Fig.4.** Simulation cycle of a single independent test

Since the arguments of these functions are either primary inputs of the circuit or the nodes supported by MISR, we can regard the set of functions (3) as a model of 5 interconnected combinational circuits, which can be fault simulated independently.

**Table 2.** A test sequence for circuit in Fig.3

| Cl | Input sequences | | Output sequences | |
|---|---|---|---|---|
| | Test $T_i$ | Test $T_{i+1}$ | Test $T_i$ | Test $T_{i+1}$ |
| 1 | $X_1^1, X_2^1, X_3^1, RES_7^1$ | | | |
| 2 | $X_1^2, X_2^2, X_3^2$ | $X_1^2, X_2^2, X_3^2, RES_7^2$ | $Z_1^2$ | |
| 3 | $RES_8^3, X_4^3$ | $X_1^3, X_2^3, X_3^3$ | $R_7^3, Z_1^3$ | $Z_1^3$ |
| 4 | | $RES_8^4, X_4^4$ | | $R_7^4, Z_1^4$ |
| 5 | $X_5^5$ | | $R_8^5, Y_1^5$ | |
| 6 | | $X_5^6$ | $Y_2^6$ | $R_8^6, Y_1^6$ |
| 7 | | | | $Y_2^7$ |

Table 2 represents two (shifted in one clock cycle) input sequences of the two test segments $T_i$ and $T_{i+1}$, and the related output sequences captured by MISR in the

test points $Z_1$, $R_7$, $R_8$, and directly at outputs $Y_1$ and $Y_2$, which can be as well fed into MISR. The table represents the simulation order of the functions (3). Because of the RULES 1 and 2 are satisfied in the modified circuit in Fig 3, the input sequences of $T_i$ and $T_{i+1}$,can be regarded as independent test patterns, spread merely over different time frames. In this way, a full test sequence applied to the circuit in Fig.3 can be split into a set of independent test segments, all shifted by one clock one after another. Since the test segments can be treated as a set of independent test patterns, they can be fault simulated by PPECPT in parallel as in case of combinational circuits.

## IV. Experimental data

As experimental results we compare in Table 3 the speed of SAF simulation in sequential circuits (where all the latches are fed into MISR) by the PPECPT method described in Section 2 with different known fault simulators for combinational circuits: FSIM [15], and two state-of-the-art commercial simulators C1 and C2 from major CAD vendors. Simulation times were calculated for 10000 patterns. Experiments were run on a 1.5GHz Ultra SPARC IV+ workstation using SunOS 5.10.

**Table 3**. Comparison of PPECPT with other fault simulation methods for circuits with full scan-path

| Circuts | Number of gates | SAF simulation time, s | | | |
|---|---|---|---|---|---|
| | | Fsim | C1 | C2 | PPECPT |
| c3540 | 2784 | 2.0 | 7.4 | 43 | 0.9 |
| c5315 | 4319 | 1.4 | 5.6 | 57 | 0.8 |
| c6288 | 4846 | 12.1 | 27.8 | 284 | 7.4 |
| s15850 | 14841 | 5.4 | 12.1 | 111 | 2.7 |
| s38417 | 34831 | 16.2 | 31.4 | 310 | 7.0 |
| s38584 | 36173 | 12.1 | 23.2 | 320 | 6.4 |
| b14 | 19491 | N/A | 49.2 | N/A | 14.5 |
| b15 | 18248 | N/A | 39.1 | N/A | 26.6 |
| b17 | 64711 | N/A | 117 | N/A | 77.8 |
| Average speed gain | | **2.0** | **4.3** | **45** | 1 |

In [16] we have presented a family of benchmark circuits which represent different architectures of a bio-impedance signal analyser (a pipe-lined signal processor) with the same functionality. We investigated the feasibility of the proposed fault simulation method for calculating the fault coverage of the at-speed functional self-test developed for these processors. The results of fault simulation for the whole family of 8 processors (column 1) are presented in Table 4 where LS denotes the behaviour level logic simulation time, FS denotes the LS multiplied by the number of faults to be simulated one by one, and the PPECPT shows the simulation time needed for the proposed method. The experiments showed that the gain we achieved by using the proposed method is around 2-3 orders of magnitude. For this advantage we have to pay by the cost of added set of MISR which however is comparable to the cost of scan-path. On the other hand, we achieve by the proposed method dramatic speed-up in the test time, compared to the scan-path approach, and improved fault diagnosis.

## V. Conclusions

In this paper we have proposed a novel approach for fault simulation in sequential circuits which allows to achieve dramatic speed-up in simulation time compared to the traditional single fault simulation in sequential circuits. The high speed is achieved thanks to removing the problem of sequential dependence of simulated signals in different time frames by improving observability of the circuit by inserting a set of MISRs at selected test points.

**Table 4**. Comparison of the proposed method with single fault simulation in sequential circuits

| Circuts | Number of faults | SAF simulation time, s | | | Gain |
|---|---|---|---|---|---|
| | | LS | FS | PPECPT | |
| 8a | 112034 | 0.155 | 17365 | 30.0 | 579 |
| 8b | 83940 | 0.152 | 12759 | 24.7 | 517 |
| 8be | 99330 | 0.168 | 16687 | 62.1 | 269 |
| 8bk | 86878 | 0.159 | 13814 | 25.2 | 548 |
| 8bs | 100820 | 0.154 | 15526 | 173.4 | 90 |
| 8c | 122386 | 0.159 | 19459 | 35.9 | 542 |
| 8d | 123012 | 0.161 | 19804 | 35.5 | 558 |
| 8de | 136876 | 0.164 | 22447 | 81.3 | 276 |

The main novelties of the paper are as follows. Instead of full scan-path we propose to use MISR for monitoring the circuit in selected test points. As a consequence, we can use instead of scan-path testing at-speed functional test which guarantees better test quality. Improved observability of the circuit allows better fault diagnostic resolution. Finally, a dramatic speed-up of fault simulation, compared to the traditional non-parallel fault simulation of sequential circuits, was achieved.

**Acknowledgement:** The work was supported by EC FP7 STREP project BASTION and by Research Centre CEBE funded by EU Structural Funds.

## References

[1] [1] J.A.Waicukauski, et.al. Fault Simulation of Structured VLSI. VLSI Systems Design, Vol.6, No.12, pp.20-32, 1985.

[2] D.B. Armstrong. A deductive method for simulating faults in logic circuits. IEEE Trans. Comp., C-21(5), 464-471, 1972.

[3] E.G.Ulrich, T.Baker. Concurrent simulator of nearly identical digital networks. IEEE Trans.on Comp.,7(4), pp.39-44, 1974.

[4] W.T.Cheng, M.L.Yu. Differential fault simulation: a fast method using minimal memory. DAC, pp.424-428, 1989.

[5] M.Abramovici, P.R.Menon, D.T.Miller. Critical Path Tracing - An Alternative to Fault Simulation. Proc. 20th Design Automation Conference, pp. 2-5, 1987.

[6] K.J.Antreich, M.H.Schulz. Accelerated Fault Simulation and Fault Grading in Combinational Circuits. IEEE Trans. On Computer-Aided Design, Vol. 6, No. 5, pp.704-712, 1987.

[7] L.Wu, D.M.H.Walker. A Fast Algorithm for Critical Path Tracing in VLSI. Int. Symp. on Defect and Fault Tolerance in VLSI Systems, Oct. 2005, pp.178-186.

[8] R.Ubar, S.Devadze, J.Raik, A.Jutman. Ultra Fast Parallel Fault Analysis on Structural BDDs. ETS, Freiburg, May 20-24, 2007.

[9] R.Ubar, S.Devadze, J.Raik, A.Jutman. Parallel X-Fault Simulation with Critical Path Tracing Technique. IEEE Conf. Design, Automation & Test in Europe - DATE-2010, Dresden, Germany, March 8-12, 2010, pp. 1-6.

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

[11] L. Bushard, N. Chelstrom, S. Ferguson, B. Keller. DFT of the Cell Processor and its Impact on EDA Test Software. In IEEE Asian Test Symposium, 2006, pp. 369-374.

[12] S. Wang, S. Gupta. ATPG for heat dissipation minimization during scan testing. In ACM IEEE Design Automation Conference, 1997, pp. 614-619.

[13] L. Chen, S. Ravi, A. Raghunathan, S. Dey. A Scalable Software-Based Self-Test Methodology for Programmable Processors. In IEEE/ACM Design Automation Confernce, 2003, pp. 548-553.

[14] Thayse, "Boolean Calculus of Differences", Springer Verlag, 1981T

[15] H.K.Lee, D.S.Ha. SOPRANO: An Efficent Automatic Test Pattern Generator for Stuck-Open Faults in CMOS Combinational Circuits. DAC, Orlando, FL, June 1990.

[16] M.Gorev, R.Ubar, P.Ellervee, S.Devadze, J.Raik, M.Min. At-Speed Self-Testing of High-Performance Pipe-Lined Processing Architectures. IEEE Conference NORCHIP, Vilnius, 2013.

# 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.
71. **Sergei Kostin.** *Self-Diagnosis in Digital Systems.* 2012.

72. **Mihkel Tagel.** *System-Level Design of Timing-Sensitive Network-on-Chip Based Dependable Systems.* 2012.
73. **Juri Belikov.** *Polynomial Methods for Nonlinear Control Systems.* 2012.
74. **Kristina Vassiljeva.** *Restricted Connectivity Neural Networks based Identification for Control.* 2012.
75. **Tarmo Robal.** *Towards Adaptive Web – Analysing and Recommending Web Users' Behaviour.* 2012.
76. **Anton Karputkin.** *Formal Verification and Error Correction on High-Level Decision Diagrams.* 2012.
77. **Vadim Kimlaychuk.** *Simulations in Multi-Agent Communication System.* 2012.
78. **Taavi Viilukas.** *Constraints Solving Based Hierarchical Test Generation for Synchronous Sequential Circuits.* 2012.
79. **Marko Kääramees.** *A Symbolic Approach to Model-based Online Testing.* 2012.
80. **Enar Reilent.** *Whiteboard Architecture for the Multi-agent Sensor Systems.* 2012.
81. **Jaan Ojarand.** *Wideband Excitation Signals for Fast Impedance Spectroscopy of Biological Objects.* 2012.
82. **Igor Aleksejev.** *FPGA-based Embedded Virtual Instrumentation.* 2013.
83. **Juri Mihhailov.** *Accurate Flexible Current Measurement Method and its Realization in Power and Battery Management Integrated Circuits for Portable Applications.* 2013.
84. **Tõnis Saar.** *The Piezo-Electric Impedance Spectroscopy: Solutions and Applications.* 2013.
85. **Ermo Täks.** *An Automated Legal Content Capture and Visualisation Method.* 2013.
86. **Uljana Reinsalu.** *Fault Simulation and Code Coverage Analysis of RTL Designs Using High-Level Decision Diagrams.* 2013.
87. **Anton Tšepurov.** *Hardware Modeling for Design Verification and Debug.* 2013.
88. **Ivo Müürsepp.** *Robust Detectors for Cognitive Radio.* 2013.
89. **Jaas Ježov.** *Pressure sensitive lateral line for underwater robot.* 2013.
90. **Vadim Kaparin.** *Transformation of Nonlinear State Equations into Observer Form.* 2013.
91. **puudub**
92. **Reeno Reeder.** *Development and Optimisation of Modelling Methods and Algorithms for Terahertz Range Radiation Sources Based on Quantum Well Heterostructures.* 2014.
93. **Ants Koel.** *GaAs and SiC Semiconductor Materials Based Power Structures: Static and Dynamic Behavior Analysis.* 2014.
94. **Jaan Übi.** *Methods for Coopetition and Retention Analysis: An Application to University Management.* 2014.
95. **Innokenti Sobolev.** *Hyperspectral Data Processing and Interpretation in Remote Sensing Based on Laser-Induced Fluorescence Method.* 2014.
96. **Jana Toompuu.** *Investigation of the Specific Deep Levels in p-, i- and n- Regions of GaAs p+-pin-n+ Structures.* 2014.
97. **Taavi Salumäe.** *Flow-Sensitive Robotic Fish: From Concept to Experiments.* 2015.
98. **Yar Muhammad.** *A Parametric Framework for Modelling of Bioelectrical Signals.* 2015.
99. **Ago Mõlder.** *Image Processing Solutions for Precise Road Profile Measurement Systems.* 2015.
100. **Kairit Sirts.** *Non-Parametric Bayesian Models for Computational Morphology.* 2015.
101. **Alina Gavrijaševa.** *Coin Validation by Electromagnetic, Acoustic and Visual Features.* 2015.
102. **Emiliano Pastorelli.** *Analysis and 3D Visualisation of Microstructured Materials on Custom-Built Virtual Reality Environment.* 2015.
103. **Asko Ristolainen.** *Phantom Organs and their Applications in Robotic Surgery and Radiology Training.* 2015.
104. **Aleksei Tepljakov.** *Fractional-order Modeling and Control of Dynamic Systems.* 2015.
105. **Ahti Lohk.** *A System of Test Patterns to Check and Validate the Semantic Hierarchies of Wordnet-type Dictionaries.* 2015.
106. **Hanno Hantson.** *Mutation-Based Verification and Error Correction in High-Level Designs.* 2015.
107. **Lin Li.** *Statistical Methods for Ultrasound Image Segmentation.* 2015.
108. **Aleksandr Lenin.** *Reliable and Efficient Determination of the Likelihood of Rational Attacks.* 2015.