TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
16/2019

Methods for Improving the Accuracy and Efficiency of Fault Simulation in Digital Systems

JAAK KÕUSAAR

TTÜ PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Computer Systems

This dissertation was accepted for the defense of the degree of Doctor of Philosophy in Computer and Systems Engineering 25.01.2019

Supervisor: Prof. Raimund-Johannes Ubar, DSc
Department of Computer Systems
Tallinn University of Technology
Tallinn, Estonia

Co-Supervisor: Prof. Jaan Raik, PhD
Department of Computer Systems
Tallinn University of Technology
Tallinn, Estonia

Opponents: Prof. Zebo Peng, PhD
Department of Computer and Information Science
Linköping University
Linköping, Sweden

Prof. Vladimir Hahanov, PhD
Department of Computer Aided Design of Computers
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine

Defense of the thesis: 26.04.2019, Tallinn

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

Jaak Kõusaar:

---
allkiri

European Union
European Social Fund

Investing in your future

ISSN 2585-6898 (publication)
ISBN 978-9949-83-407-5 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9949-83-408-2 (PDF)

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
16/2019

Meetodid digitaalsüsteemide rikete simuleerimise täpsuse ja efektiivsuse tõstmiseks

JAAK KÕUSAAR

TTÜ KIRJASTUS

To feel the value of success, you have to sense the failure. To truly feel it, you have to fail so bad that there seems to be no way out of it without somebody offering you the helping hand. And then… there is no greater feeling of success, when you deny it and help yourself!

But usually it's just wiser to take the damn offer…

Table of Contents

List of Publications

The work of this thesis is based on the following publications:

I R. Ubar, J. Raik, S. Kostin, J. Kõusaar. Multiple Fault Diagnosis with BDD based Boolean Differential Equations. Proc. of Baltic Electronics Conference, Tallinn, October 3-5, 2012.

II J. Kõusaar, R. Ubar. About testing of transition delay faults. Proceedings of the Workshop BELAS, Cottbus, Germany, 2014.

III 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, 2014.

IV J. Kõusaar, R. Ubar. 7-Valued Algebra for Transition Delay Fault Analysis. Proceedings of Baltic Electronics Conference - BEC, Tallinn, October 6-8, 2014.

V J. Kõusaar, R. Ubar, I. Aleksejev. Complex Delay Fault Reasoning with Sequential 7-valued Algebra. Proceedings of 16th IEEE Latin American Test Symposium - LATS, Puerto Vallarta, Mexico, March 25-27, 2015.

VI R. Ubar, J. Kõusaar, S. Devadze, J. Raik. Transition Delay Fault Simulation with Parallel Critical Path Back-Tracing and 7-valued Algebra. Microprocessors and Microsystems - MICPRO, Elsevier, Volume 39, Issue 8, Nov. 2015, pp. 1130-1138.

VII R. Ubar, J. Kõusaar, M. Gorev, S. Devadze. Combinational Fault Simulation in Sequential Circuits. Proceedings of International Symposium on Circuits and Systems - ISCAS, Lisbon, Portugal, May 24-27, 2015.

VIII J. Kõusaar, R. Ubar, S. Kostin, S. Devadze, J. Raik. Parallel Critical Path Tracing Fault Simulation in Sequential Circuits. 25th International Conference Mixed Design of Integrated Circuits and Systems - MIXDES, June 21-23, 2018, Gdynia, Poland.

IX J. Kõusaar, S. Kostin, R. Ubar, S. Devadze, J. Raik, “Exact Parallel Critical Path Fault Tracing to Speed-Up Fault Simulation in Sequential Circuits”, *International Journal of Microelectronics and Computer Science*, vol. 9, pp. 9-18, October 2018.

Author's Contributions to the Publications

Contributions to the publications in this thesis are:

I The author contributed in verification of the 5-valued algebra, investigated the feasibility of using the algebra in modeling the diagnostic process and carried out experiments. The author also helped to elaborate the manuscript, and presented the paper.

II The author was the co-developer of the algorithm of delay fault simulation and implementor of a software based on that algorithm. The author conducted experiments, carried out analysis of experimental results, wrote the manuscript and presented the paper at the conference.

III The author contributed in developing the 7-valued algebra, developed and implemented four different types of delay fault simulation, and conducted the experiments. Author also contributeds in writing the paper, and presented it in the conference.

IV The author carried out research on verification of the 7-valued algebra, implemented it into the delay fault simulation algorithms, investigated the feasibility of using the new algebra for delay fault coverage calculation. He also conducted experiments, wrote the paper and presented it at the conference.

V The author was a co-developer of the full algorithm of the transition delay fault simulation, and implementor of a software based on that algorithm. He also carried out experiments, and contributed in writing the paper.

VI The author developed a method of combining two types of simulation targeting high transition delay fault coverage, created a set-up for carring out experiments, and investigated the effectiveness of the new delay fault simulation approach. He also contributed in writing the paper.

VII The author was a co-developer of the new approach of fault simulation in sequential circuits, being responsible for converting sequential fault simulation task into the combinational one. He contributed in creating the set-up of experiments, carried out simulations to compare the approach with classical sequential method. He wrote also a part of the paper.

VIII The author developed the algorithm of classification of faults to be simulated in parallel and sequentially. He contributed in conducting the experiments, wrote the second part of the manuscript, and presented the paper at the conference.

IX The author investigated and explained the the problems which make parallel critical path tracing in sequential circuits difficult. He combined two methods to be used in a single algorithm of fault simulation in sequential circuits. He contributed in conducting the experiments, and in writing the paper.

Abbreviations

ADD	Algebraic Decision Diagram
BDD	Binary Decision Diagram
CAD	Computer-Aided Design
CSAF	Conditional Stuck-At Fault
DFT	Delay Fault Testing
DSSBDD	Diagnostic Structurally Synthesized Binary Decision Diagram
EVBDD	Edge-Valued Binary Decision Diagram
FDD	Functional Decision Diagram
FFR	Fan-out Free Region
ITRS	International Technology Roadmap for Semiconductors
MDD	Multi-valued Decision Diagram
MISR	Multiple Input Signal Register
MSAF	Multiple Stuck-At Fault
MTBDD	Multi-Terminal Binary Decision Diagram
PCPT	Parallel Critical Path Tracing
PDF	Path Delay Fault
PPECPT	Parallel Pattern Exact Critical Path Tracing
PPSFP	Parallel Pattern Single Fault Propagation
ROBDD	Reduced Ordered Binary Decision Diagram
SAF	Stuck-At Fault
SDF	Segment Delay Fault
SSBDD	Structurally Synthesized Binary Decision Diagram
SSFS	Slow Sequential Fault Simulation
TDD	Ternary Decision Diagram
TDF	Transition Delay Fault
TP	Test Pattern
TPDF	Transition Path Delay Faults
VLSI	Very-Large-Scale Integration
ZSBDD	Zero Suppressed Binary Decision Diagram

Introduction

The general focus of the thesis is fault simulation in digital circuits. As the problem of fault simulation, in general, is very large, the research in Thesis is concentrated on developing new applications for the two existing simulation concepts:

1. Critical path tracing as a very fast fault simulation concept used so far only for combinational circuits, and not for sequential circuits.
2. Structurally Synthesized BDDs (SSBDD) used so far for fault simulation and test generation, but not for fault diagnosis.

The more specific narrow focus of the thesis is to develop new fault simulation methods, supported by the mentioned two concepts, and targeting delay fault simulation, fault simulation in sequential circuits, and diagnostic fault simulation.

The introduction gives an overview of state-of-the-art in the area addressed in the thesis. The motivation of research is given, followed by description of the research objectives and formulation of the problems to be solved with underlying the main goals of the research. Thereafter, the main contributions are outlined, and the overall structure of the thesis is described.

Motivation – improving the speed and accuracy of fault simulation in digital circuits

The growing complexity of electronic designs, resulting from the progress and developments in emerging nanoelectronic technology, requires increase of the design productivity and speed-up of the design and test tools to cope with ever strengthening requirements of time to market and of the quality of today's electronic products. Logic-level simulation is one of the most critical procedures in digital design for verification purposes, fault simulation, test generation and fault diagnosis. This makes it a critical issue affecting the overall cost and quality of electronic design projects [1]-[4].

With constantly increasing speed of nanoelectronic circuits, violations of the performance specifications have become a major factor affecting the product quality level, and have forced intensive research in the field of testing delay faults, which is also the topic of the current thesis.

Another important topic is fault simulation in sequential circuits, which due to the presence of memory do not allow as fast fault simulation as in combinational circuits. To increase the speed of fault analysis in sequential circuits due to the growing complexity of circuits is one of the most important challenges in the test field. This has motivated to revisit the traditional methods and algorithms of fault simulation, to discover and develop new ideas and making the existing concepts and approaches more efficient to increase the performance of fault simulation in sequential circuits.

The efficiency and speed of simulation highly depends on the data structures used in simulation procedures and on the way, how these procedures are organized. Binary Decision Diagrams (BDD) has become state-of-the-art data structure in the VLSI CAD tools for representation and manipulation of Boolean Functions [5]-[7]. Recently, it was shown that a subclass of Structurally Synthesized BDDs (SSBDD) opens new possibilities to reduce the overall model structural complexity compared to the traditional gate level circuit representations and more accurately reflect the structural properties of digital circuits [8]-[10].

These findings motivated to carry out research and investigations of using SSBDDs to introduce improvements into current approaches to fault simulation, especially targeting the special class of delay faults, the general class of sequential circuits, and diagnostic purposes of simulation.

Research objectives and problem formulation

The goal of testing digital circuits is to make sure that the circuit is working correctly. In other words, the goal is to check whether no fault is present in the circuit, or if the circuit is found faulty, to diagnose the fault location. Fault simulation is the procedure of measuring the quality of a given test in terms of fault coverage.

There are a lot of different fault models used for measuring the fault coverage like Stuck-at Faults (SAF) [1], conditional SAF (CSAF), bridging faults (or shorts), delay faults etc. [1]-[4]. Traditionally, only single faults in digital circuits are considered and simulated. Due to the huge size of possible combinations of multiple faults in circuits they are traditionally neglected, which is the general disadvantage of the current state-of-the-art in digital test. CSAF fault class is an extension of SAF faults, where each SAF is accompanied by additional logic conditions (constraints) to model as exactly as possible the real physical defects. Bridging faults and delay faults are examples of CSAF.

Delay faults is a special fault class (extension of CSAF) into the sequential test domain, and can be considered as the most general fault class for digital circuits [11]. To cope with the generality, and complexity of handling the delay fault class, several subclasses of delay faults are introduced: path delay faults (PDF), segment delay faults (SDF), transition delay faults (TDF), a. o. with different tradeoffs between the size of the fault model and accuracy of covering the real physical defects. In this thesis, investigations are presented which will improve the state-of-the-art of delay fault modeling efficiency.

There are a lot of fault simulations methods developed for combinational circuits [1], among of them the analytical approaches like deductive simulation and critical path tracing of faults are the most efficient ones, which so far have not been found applicable for sequential circuits. In this thesis, research results are presented which fill up the mentioned gap.

SSBDDs have been shown as a promising data structure for modeling digital circuits in a uniform way at both, gate and macro (subcircuit) levels, providing in this way [10] an excellent possibility for hierarchical handling of circuits to speed-up simulation. Other two important properties of SSBDDs are the inherent fault collapsing (fault model compaction) and direct mapping between the gate-level faults and the structural entities (nodes) of SSBDDs [12]. In this thesis, a novel application field of SSBDDs is introduced which allows formally model the processes of multiple fault diagnosis.

The following problems as research objectives of the thesis were formulated with the goal to advance state-of-the-art in the field of fault simulation in digital circuits:

- to advance existing delay fault models with the goal of improving the accuracy of evaluation of delay fault testing quality (fault coverage) in digital circuits,
- to create methods and algorithms for delay fault simulation in digital circuits to speed-up the calculation of delay fault coverage,
- to improve the performance of fault simulation in sequential circuits by making parallel critical path tracing method, used so far only for combinational circuits, applicable also for sequential circuits, and

- to develop a novel SSBDD-based formal approach for modeling faults and simulating fault diagnosis procedures, the first time, for the general case of multiple faults in digital circuits.

Thesis contributions

In the thesis, the following new results in the research field of fault simulation of digital circuits have been achieved.

1. A new transition delay fault (TDF) model (non-robust functionally sensitized TDF) and a new method for simulating delay faults were developed, which allow to improve the quality of delay fault testing, compared to state-of-the-art.
2. For TDF fault reasoning, a novel 7-valued algebra was developed, which allows TDF analysis concurrently at different fault sensitizing conditions (robust, non-robust, functional and non-robust functional).
3. A new method for parallel critical path tracing (PCPT) fault simulation for sequential circuits was proposed, which is based on converting a sequential circuit into combinational one by using multiple input signature registers for cutting the feedback loops. The method allows considerable speed-up of simulation, compared to traditional approaches.
4. To achieve the high speed-up of simulation, the method of PCPT based fault analysis was extended to be used in sequential circuits.
5. A new method of fault simulation for sequential circuits was proposed, which is based on the PCPT fault simulation, but does not need explicit cutting of feedback loops. The method allows considerable speed-up of simulation, compared to traditional approaches.
6. A new type of Diagnostic SSBDDs (DSSBDD) is introduced, and as such, a new application field is introduced for using SSBDDs in fault simulation purposes. The first time, a method is proposed for fault diagnosis in digital circuits in a general case of multiple fault assumption by solving Boolean differential equations. For that purpose, DSSBDDs are proposed for simulating diagnostic experiments, and a special 5-valued algebra was developed, for generating fault diagnosis statements.

Thesis structure

The Thesis consists of 6 chapters, which present the carried out research, and contribute with new research results in the following way.

Chapter 1 presents the background and overview of the fault simulation methods in digital circuits. First the concept of modeling digital circuits with SSBDDs and its main applications are described. Second, the concept of fast parallel critical path tracing of faults in combinational circuits is presented. On the basis of these concepts the main tasks of Thesis are formulated. Thereafter, overviews of different existing fault simulation approaches are discussed. The background description concentrates on the delay fault simulation, on the problems of fault simulation in sequential circuits, and on the problem of multiple fault diagnosis, with showing the positions of the contributions of Thesis.

Chapter 2 is devoted to development of a new advanced method of delay fault simulation. First a new type of transition delay fault (TDF) model is introduced to improve the accuracy of measuring the delay fault coverage. Two novel algebras of delay fault reasoning and a new overall delay fault simulation procedure are presented for calculating the fault coverage with improved accuracy. Experimental data are presented, which demonstrate the advantages of the new methods compared to state-of-the-art.

Chapter 3 presents a new method for fault simulation in sequential circuits, where instead of traditional fault-by-fault sequential simulation, a parallel fault simulation method is developed, which is based on reasoning concurrently of all the faults in the circuit, using the critical path tracing concept. To make this concept, originally developed for combinational circuits, applicable also for sequential circuits, a simple update of a design is needed (design-for-testability) to cut the global feedback loops by introducing a multiple input signature register (MISR). The one-cycle combinational critical path tracing algorithm is then generalized to multi-cycle sequential critical path tracing algorithm applicable for sequential circuits without global feedback loops. Experimental results are presented which demonstrate dramatic speed-up of simulation, compared to traditional approaches.

In Chapter 4 a new method for fault simulation in sequential circuits is proposed, where no updates in the design are needed, and the global feedback loops of the circuit are cut algorithmically. In the proposed approach, two methods are combined: fast parallel critical path tracing, and slow sequential fault-by-fault simulation. The set of all faults is partitioned algorithmically into two subsets, so that one of them can be simulated by the fast critical path tracing, and the rest of faults remain to be analyzed by the slower sequential fault-by-fault simulation. Experimental results show the speed-up of the new method compared to state-of-the-art.

In Chapter 5, a new type of Diagnostic SSBDDs (DSSBDD) is introduced, and as such, a new application field is introduced for using SSBDDs in fault simulation purposes. A novel idea is proposed for representing Boolean differential equations in form of SSBDDs for simulating fault diagnosis processes in the general case of the multiple fault assumption (in the traditional practice, single fault assumptions are made). A method is proposed for converting Boolean differential equations as a model of diagnostic test experiment into SSBDDs where each node represents a signal transition as a faulty case. An algorithm was proposed to simulate the diagnostic experiments by manipulations of SSBDDs, where for manipulation of SSBDDs, a special 5-valued algebra was developed. A novel hierarchical approach for fault diagnosis of digital circuits is developed, where the role of the proposed SSBDD based diagnostic simulation method is to specify the fault location in ambiguous situations due to self-masking of multiple faults. Experimental results conclude the chapter.

Chapter 6 draws conclusions from the thesis and outlines the directions for future work.

1 Background

Chapter 1 presents the background and overview of the fault simulation methods in digital circuits. First the concept of modeling digital circuits with SSBDDs and its main applications are described. Second, the concept of fast parallel critical path tracing of faults in combinational circuits is presented. On the basis of these concepts the main tasks of Thesis are formulated. Thereafter, overviews of different existing fault simulation approaches are discussed. The background description concentrates on the delay fault simulation, on the problems of fault simulation in sequential circuits, and on the problem of multiple fault diagnosis, with showing the positions of the contributions of Thesis.

1.1 Structurally Synthesized BDDs and fault simulation

The BDDs for representing Boolean functions were the first time proposed by Lee [13], however, this technique was not popular among the researchers for a long time due to very rapid growth of the complexity of BDDs in case of large Boolean functions. In 1986, a new data structure, called reduced ordered BDDs (ROBDDs) was proposed by Bryant [5], by showing the simplicity of the graph manipulation and proving the model canonicity. This work made BDDs one of the most popular representation models of Boolean functions [6], [14]-[15]. Many different types of BDDs have been proposed since and investigated during decades, including shared or multi-rooted BDDs [16], ternary decision diagrams (TDD), multi-valued decision diagrams (MDD) [17], edge-valued binary decision diagrams (EVBDD) [16], functional decision diagrams (FDD) [18], zero-suppressed BDDS (ZBDD) [19], algebraic decision diagrams (ADD) [20], free BDDs [21], multi-terminal BDDs (MTBDD) and hybrid BDDs [22] etc. More complete overviews about different types of BDDs can be found in [7], [14]-[15].

As mentioned above, BDDs have been traditionally used to represent the Boolean functions. Much less attention has been drawn for modeling the structural properties of digital circuits with BDDs, like gates, connections between gates and signal paths of the related circuits. These aspects of logic circuits were first introduced into BDDs in [8]-[9], [23], where one-to-one mappings between the nodes of BDDs and signal paths in the related gate-level circuits were introduced. Such BDDs were initially called alternative graphs [8]-[9], and later structurally synthesized BDDs (SSBDDs) [23] to put the accent on the way how the BDDs were constructed directly from the gate-level circuits.

Based on the one-to-one mapping between the nodes in SSBDDs and the signal paths in circuits, efficient method of critical path tracing [24] fault simulation was developed. Later, a very fast fault simulation approach based on parallel reasoning of faults on SSBDDs simultaneously for many test patterns was developed in [25], and later generalized for extended fault classes like conditional SAF [26] and X-fault model [27]. SSBDDs have been used for optimizing fault localization processes in digital circuits [28], for design error diagnosis [29], for testability evaluation of circuits [30], and for optimization of SSBDDs for fast evaluation of the quality of Built-in Self-Test of digital systems [31].

$$y = (x_{11}x_{12}) \vee (x_{12}x_{31}x_4) \vee (\overline{x_{13}}x_{22}x_{32}) \quad (1\text{-}1)$$

An example of a SSBDD for a digital circuit, described also by a Boolean function (1-1) is presented on Figure 1.1

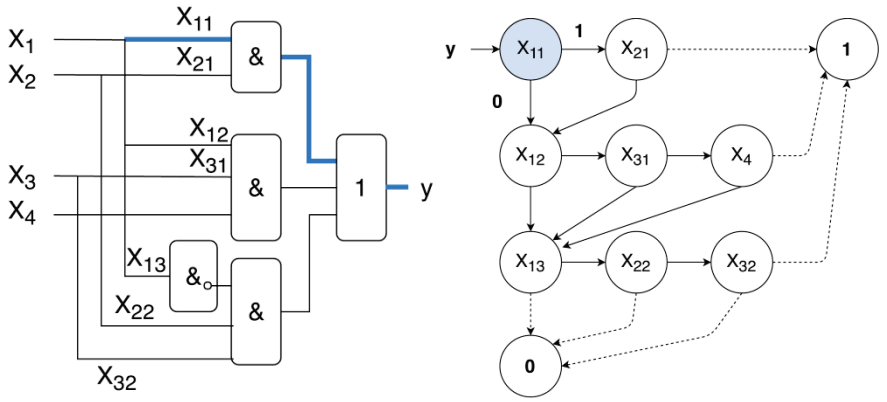


Figure 1.1: A logic circuit and its SSBDD

By convention, the right-directed edges from a node in SSBDD correspond to the value 1, and the down-directed edges correspond to the value 0 of the node variable. The node variables in a SSBDD may be also inverted. The edge $1(0)$ of a node m with node variable $x(m)$ is considered activated if the value of $x(m)$ is $1(0)$. A path l between the nodes $x_i$and $x_j$is called activated if all the edges which form the path l are activated.

Such graph is derived directly from the circuit (or from the formula (1-1)) and represents the structure of the circuit (Figure 1.1), hence, the term structurally synthesized BDD. There exists a one-to-one mapping between the nodes in the graph and the inputs of the related fan-out free region (FFR) of the circuit in Figure 1.1 (or the literals in the formula (1-1)). For example, the node x_{11} in SSBDD represents the signal path from x_{11} to y in the circuit.

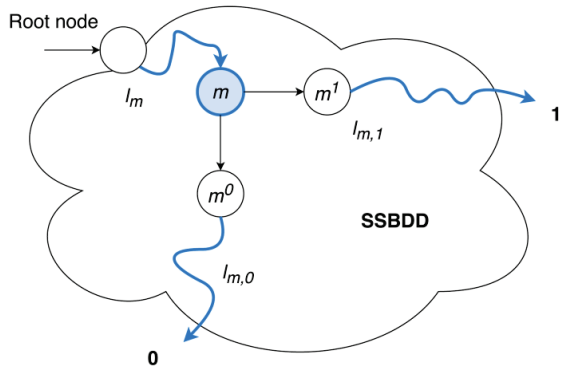


Figure 1.2: Illustration of fault simulation and test generation in SSBDDs

The advantage of SSBDDs is in reducing the complexity of modeling the gate-level circuits. For example, in SSBDDs only the inputs of the related FFRs are represented (the internal nodes of FFRs are collapsed in SSBDDs). Since each SSBDD node represents a path of FFR, then all the stuck-at-faults (SAF) along the path in FFR are collapsed into only two SAF faults of the related node in SSBDD.

SSBDDs have been used for both fault simulation and for test pattern generation. In Figure 1.2 the conditions to be satisfied in SSBDD for both operations are illustrated. The fault at the node *m* can be detected by a test pattern TP if the three paths highlighted with blue arrows in Figure 1.2, are activated. Fault simulation process is carried out by checking if the given test pattern is activating paths mentioned above. Test generation, in turn, means a choice of proper values for the node variables on these paths, so that the paths were activated. Efficient algorithms and procedures for fault simulation and test generation, also for solving other testing related tasks, have been developed for SSBDDs in [9], [23]-[31].

The model of SSBDDs is used in Chapters 2-4 for speeding up fault simulation, and in Chapter 5 for introducing a new application field for fault diagnosis in digital circuits. In the latter case, SSBDDs are used for representing Boolean differential equations, whereas the traditional use of BDDs is representing of Boolean functions.

1.2 Fault simulation methods for combinational circuits

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, as well as fault diagnosis relies heavily on the performance and speed of fault simulation in particular.

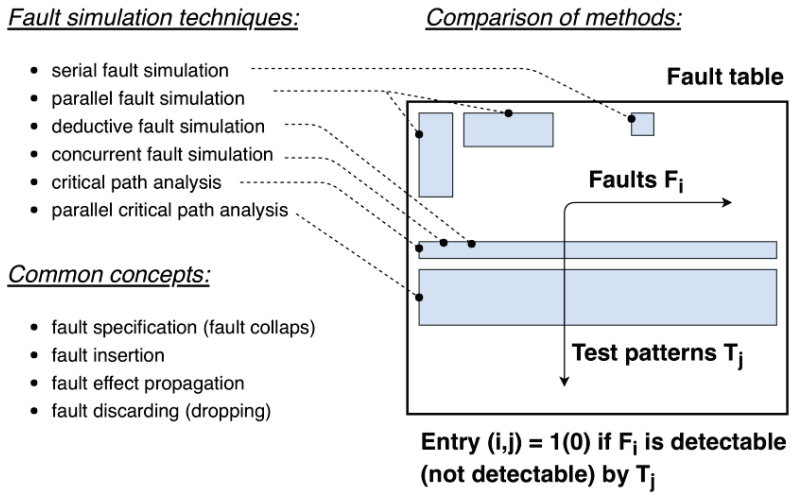


Figure 1.3: Comparison of different fault simulation methods

Such dependence is growing with the size of circuits, and hence, the scalability of the fault simulation algorithms is crucial for all similar tasks. Accelerating the fault simulation would consequently improve the performance of all the above-mentioned applications.

In Figure 1.3 an overview is given about different fault simulation techniques by showing also the productivity of every method in terms of covering the fault table with detected faults on a single run of the simulator.

Parallel pattern single fault propagation (PPSFP) concept [32] 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 [33], critical path tracing [34]-[35], stem region [36] and dominator concept [36]-[37]. These techniques have helped to reduce simulation time further.

Another trend in fault simulation methods, based on reasoning (deductive [38], concurrent [39] and differential simulation [40]) used to be very powerful because of the ability to allow to collect all detectable faults by a single run of the given test pattern. On the other hand, what they cannot do, is to produce reasoning for many test patterns in parallel.

The critical path tracing method [34]-[35] eliminates explicit fault simulation for faults within Fan-out-Free Regions (FFR).

Consider a combinational circuit as a network of fan-out free regions (FFRs), where each of the FFRs can be represented as a Boolean function $y = F(x_1, x_2, \ldots, x_n) = F(X)$, with $X = (x_1, x_2, \ldots, x_n)$ as input vector. Such a network of 5 FFRs is presented in Figure 1.4.

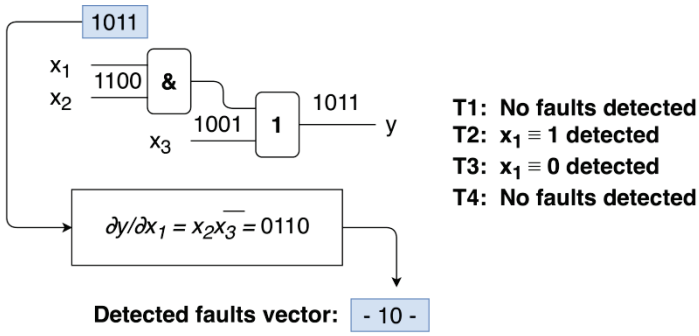


Figure 1.4: Parallel fault simulation for an FFR at the gate level

The fault simulation for a FFR, according to the 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 as illustrated in Figure 1.3.

Extension of such a fault simulation beyond the fan-out stems, i.e. beyond the FFRs is not straightforward. The easiest way would be to simulate the faults of fan-out stems separately to check if the faults inside of FFRs will propagate to the primary outputs of the circuit or not. If yes, then the critical path tracing can be continued in the related 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, was proposed in [41]. However, the rule based strategy does not allow parallel analysis of patterns, i.e. parallel rule check of many patterns simultaneously.

This drawback was removed in [42] by introducing a novel concept of Parallel Pattern Exact Critical Path Tracing (PPECPT) which can be applied efficiently also beyond FFRs. In [27], 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 order to extend the parallel critical path tracing beyond an FFR described by a function $y = F(x_1, x_2, \ldots, x_k)$, where the variable x_i, $i = 1,2,\ldots,k$, represent the end nodes of paths which fan out from a joint node x, a formula depicted in Figure 1.5 can be used.

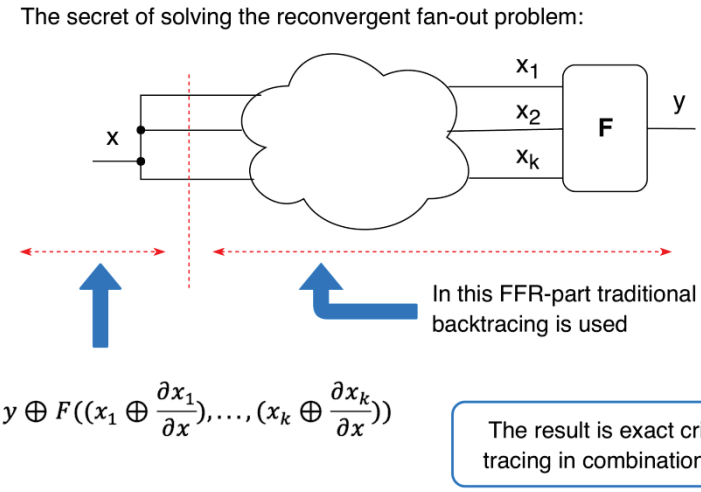


Figure 1.5: Parallel fault simulation beyond the FFR

To generalize this approach to any arbitrary network of FFRs, the concept of Boolean differentials can be used [43]-[45], as illustrated in Figure 1.5.

Consider the full Boolean differential for an FFR with a function $y = F(X)$ as

$$dy = y \oplus F\big((x_1 \oplus dx_1), \ldots, (x_n \oplus dx_n)\big) = y \oplus F(X \oplus dX) \tag{1-2}$$

Here, dx denotes the change of the value of x because of a fault at x, whereas $dy = 1$ corresponds to the case when an erroneous change of the values of arguments of the function (1-2) 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 [27] it was shown that from the expression (1-2), the following relationship can be derived:

$$\frac{\partial y}{\partial x} = y \oplus F\left(\left(x_1 \oplus \frac{\partial x_1}{\partial x}\right), \ldots, \left(x_n \oplus \frac{\partial x_n}{\partial x}\right)\right) = y \oplus F\left(X \oplus \frac{\partial X}{\partial x}\right) \tag{1-3}$$

Or, in the vector form as

$$\frac{\partial y}{\partial x} = y \oplus F\left(X' \oplus \frac{\partial X'}{\partial x}, X''\right) \tag{1-4}$$

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.

The formula (1-4) can be used for calculating the impact of the fault at the fan-out stem x on the output y of the given convergent fan-out region by consecutive calculating of Boolean derivatives over the related FFR chains starting from x up to y. For that

purpose, for each fan-out stem, which has convergence, the corresponding formulas like (1-4) should be constructed for all FFRs involved in the convergence. In the case of nested convergences, the formulas will have as well as nested structure. All these formulas will constitute a compiled partially ordered computation model for fault simulation, which can be composed by the topological analysis of the circuit [27].

In this thesis, the described very fast parallel fault simulation approach will be used for two goals. First, in Chapter 3 a novel delay fault simulation method is developed, where the described above parallel SAF fault simulation is combined with special delay fault reasoning procedures. In Chapter 4, the described above fault simulation method applicable only for combinational circuits will be extended for using it also for fault simulation in sequential circuits without global feedback loops. The problem of cutting feedback loops will be solved by introducing special design for testability procedure. In Chapter 5, the described above fault simulation method is combined with fault-by-fault simulation to make it applicable also for sequential circuits without the need of redesign for testability.

1.3 Delay fault simulation methods

With the ever increasing speed of integrated circuits, violations of the performance specifications are becoming a major factor affecting the product quality level [14]. The need for testing timing defects is further expected to grow in the nanoelectronic era.

Delay testing is used to ascertain that manufactured digital circuits meet their timing specifications. Delay faults can be modeled in different ways, using line, gate, transition, segment and path delay models. An overview of different delay fault models is presented in Figure 1.6. In the circuit a path highlighted with red bold lines is under test which consists of two test patterns. The delay fault related to this path is detected when the signal transition will be late at the moment when the clock is applied.

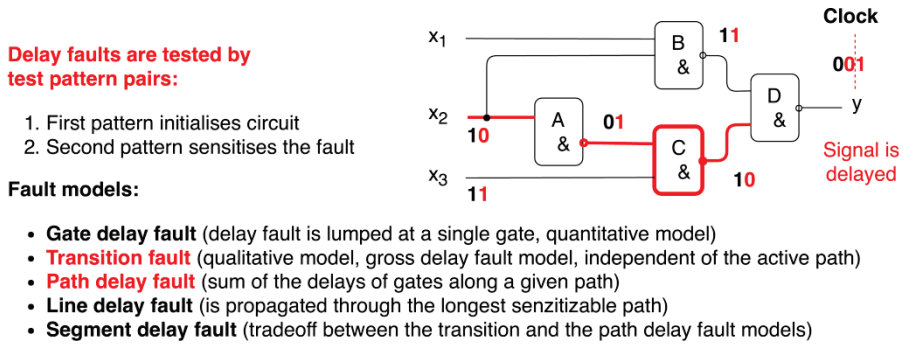


Figure 1.6: Testing of timing defects with different delay fault models

Most common are transition delay fault (TDF) model [46], and path delay fault (PDF) model [47]. The TDF captures large delay defects that affect single locations in the circuit whereas PDF captures small delays, such that each one by itself may not cause the circuit to fail, but their cumulative effect along a path from inputs to outputs may result in faulty behavior. The TDF is as well used as a logic model for stuck-open faults in CMOS circuits, which either suppress or delay the occurrence of certain transitions [48].

International Technology Roadmap for Semiconductors (ITRS) suggests the future development of methods mainly targeting small delay defects.

The main advantage of the TDF model is the linearity of the number of faults in terms of the number of connections in gate-level circuits. Another advantage is that the stuck-at-fault (SAF) test generation and fault simulation tools can be directly used for handling TDFs. The disadvantage of TDFs is that the expectation that the delay fault in a single location is large enough for propagating up to the observation points might not be realistic. On the other hand, the disadvantages of the PDF model is that the number of paths in circuits is huge, the number of robustly testable paths is in general very small, and as shown in [49], the non-robust tests may not detect delays in certain situations where the cumulative effect of small delays would be sufficient to be tested robustly.

There are attempts to extend the TDF and PDF models to exploit their advantages and alleviate the impacts of their disadvantages. The double TDF model [49] addresses the case where the increased delay of a single line is too small to cause timing fault, however, two consecutive transitions through two faulty lines would increase the probability of causing a delay behavior of the circuit. The segment fault model [51]-[53] is a restriction of the PDF model, where, instead of paths sub-paths (or segments) are used.

In [49], [54] a novel Transition Path Delay Fault (TPDF) model was proposed where the ideas of TDF and PDF models are combined to improve the capability of TDFs to detect the delays by combining tests for TDFs along the target paths. The model was proposed to capture the behavior of both small and large delay defects in a single fault model. Similarly to the conventional PDF, a test for a TPDF is required to propagate a transition from the source of the path through the path. The model additionally requires that all the transition faults along the path would be detected by a single test. The longer a path is, the smaller an extra delay on the path needs to be in order to cause the path to fail. Therefore, tests that propagate transition faults through longer paths can detect smaller delay defects.

In [55] a new TDF model is proposed called As Late as Possible Transition Fault (ALAPTF). The model aims at detecting cumulatively smaller delays, which will be missed by both the traditional TDF and the PDF models. The model makes sure that each transition is launched as late as possible at the TDF site, accumulating the small delay defects along its way.

These approaches [49] and [55] are based on the idea of creating paths along which the TDFs are consecutively tested. Both approaches require continuously activated paths or segments of paths along which the TDF will robustly propagate. In general, such paths may be missing, or it is difficult to create paths along which all the TDFs are detected. In these cases, it may be reasonable to alleviate the constraints on consecutive propagation of TDFs robustly along the tested path, so that in some gates or segments the TDFs may propagate non-robustly or along discontinuous paths.

In [49], it was shown that a test for TPDF corresponds to a type of strong non-robust test [56] for a conventional PDF associated with the same path, however, with the difference that the TPDF test additionally detects the TDFs on every line of the path. On the other hand, a strong non-robust test satisfies the weak non-robust propagation conditions [56], but not the opposite. As shown in [57], this results in a large gap in the number of detectable faults between TPDFs as defined in [49] and conventional PDFs when the weak non-robust propagation conditions are used for them. To bridge this gap, in [57], the detection conditions for TPDFs used in [49] were extended with hazard-based detection conditions for transition faults defined in [58]. When the hazard-based

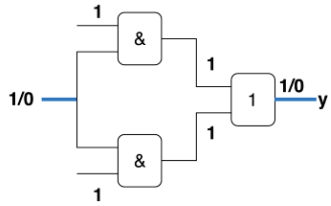
detection conditions are allowed for the transition faults included in a TPDF, it is not necessary to create transitions on every line of the path; rather it is also possible for lines on the path to have hazards. When some of the TDFs associated with a TPDF are detected under the hazard-based detection conditions, the resulting test will be a type of weak non-robust test.

In addition to the hazard-based conditions, the Thesis alleviates further conditions of TDF detection, introducing functional sensitization [56] into the detection conditions to handle multiple TDFs and adding a new type of action that is called non-robust functional sensitization condition, allowing the detection of multiple TDFs in hazard-based conditions.

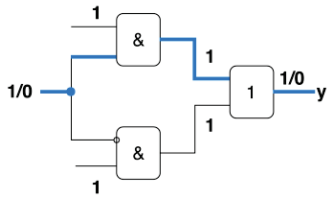
1.4 Fault simulation in sequential circuits

It was already stressed that one of the most important tasks in digital circuit design and test is simulation of faults in digital circuits. Several aspects of the quality of circuits like efficiency of test quality and dependability evaluation, test generation and fault diagnosis etc. relies heavily on the speed of fault simulation. Therefore, accelerating of fault simulation procedures would have a strong impact to all of these mentioned applications.

In Section 1.2 an overview was given about the different methods of faults simulation in combinational circuits with comparison of their efficiency. The methods of single fault simulation and parallel pattern single fault simulation are applicable also for sequential circuits.



(a) The critical path is not continuous



(b) The critical path breaks on the fan-out

Figure 1.7: Problems with critical path tracing in combinational circuits

Further, it was shown that the analytical trend based on the fault reasoning (deductive [38], concurrent [39] and differential simulation [40]) is more powerful, since these methods allow to collect all detectable faults by a single simulation run for the given test pattern. The disadvantage of these methods is that they can be applied for analysis of only a single test pattern.

Another analytical fault simulation method, called critical path tracing, filled-up this gap [34] to allow parallel fault simulation, but still for only a specific tree-like class of subcircuits, called Fan-out-Free Regions (FFR) of combinational circuits.

Figure 1.7 demonstrates that in case of (a), no critical paths exist in the FFR-part for the given test pattern, but the fault of the fan-out stem is still detected. This means that the critical paths in general case of combinational circuits are not continuous. On the other hand, in case of (b), the critical path is breaking off in the fan-out stem for the given test pattern. Which concludes that for using the concept of critical path fault tracing in arbitrary combinational circuits, dedicated rules should be introduced, taking into account the specific properties of circuits. Such rules were introduced in [41] to extend the critical path method for using in arbitrary combinational circuits. However, the use of these rules is possible only for a single pattern, which excludes the possibility of parallel fault simulation beyond the FFRs.

Recent developments [27], [42], [83] showed that the analytical approach of critical path tracing can be applied not only for tree-like FFR-s, but also for arbitrary combinational circuits with nested re-converging fan-outs, and also in parallel for many test patterns. The ideas of these methods were explained in Section 1.2.

Unfortunately, the parallel critical path tracing based fault simulation method [27], [42], [83] cannot be applied for sequential circuits due to the global feedbacks.

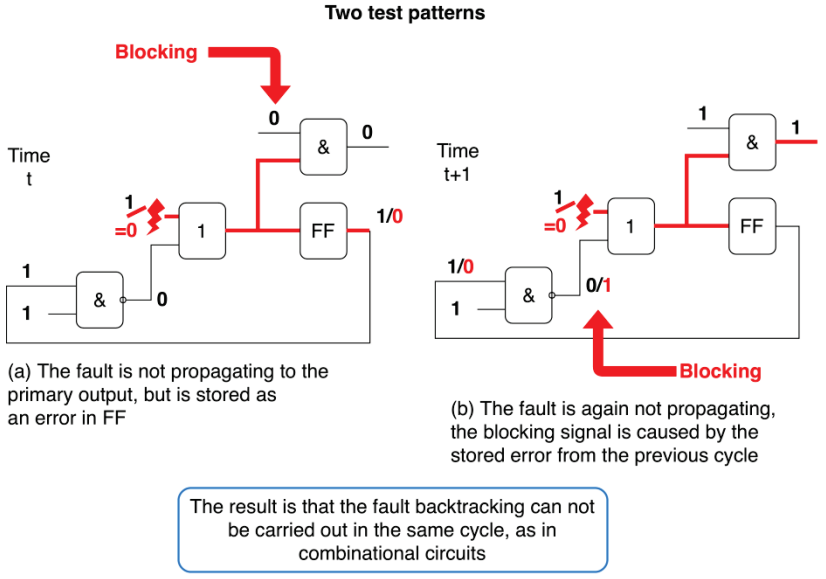


Figure 1.8: Critical path tracing in a sequential circuit

In Figure 1.8, a sequential circuit with a stuck-at fault (SAF-0) is shown. Two test patterns at the time moments t and $t+1$ are applied: $T_1 = (110)$ in Figure 1.8 (a), and $T_2 = (111)$ in Figure 1.8 (b), respectively. The simulated values for the correct circuit are shown with black numbers, and for the case of fault with red values. Critical paths activated by the patterns are shown with red bold lines. Figure 1.8 (b) displays that the fault propagating along the critical path to the output and hence, is detected. However, in the previous time moment t, in Figure 1.8 (a), the fault is propagating to the flip-flop T, and changing its state. As the result, the fault will be masked during the test pattern T_2.

Hence, the critical path in Figure 1.8 (b) is not valid due to the faulty signal from the flip-flop, and the fault will be not detected.

The example, illustrated in Figure 1.8, demonstrates that in sequential circuits, due to the feedback loops, each fault must be simulated separately, and the classical fault independent critical path analysis used in combinational circuits is not possible. As the result, the fault simulation in sequential circuits cannot perform as fast as in combinational circuits.

In the Thesis (Chapter 3), a method is proposed to fill up this gap in the following way. All the feedback loops are partitioned into two classes: global and local feedback loops. The influence of global feedbacks will be eliminated by introducing related observation possibilities (multiple input signature analyzers - MISR) which involves the need of redesign of the circuit for testability. To overcome the problem of local feedback loops (e.g. flip-flops), a modified multi-clock critical path fault tracing method is developed.

In Chapter 4, a novel method, which does not need the redesign for testability, is developed, which is based on combining the critical path tracing with traditional separate fault simulation (fault-by-fault) and, as the result, drastically increases the speed of fault analysis in sequential circuits.

1.5 Multiple fault diagnosis in digital circuits

Due to the high density of nanoscale circuits, a manufacturing defect may result in a fault involving more than one line, and the ability of the single stuck-at-fault (SSAF) model to represent physical defects decreases considerably. On the other hand, targeting multiple SAF (MSAF) as test generation objectives is infeasible, since an *n*-line circuit may have $3^n - 1$ faulty situation compared to $2n$ faulty situations under the SSAF model.

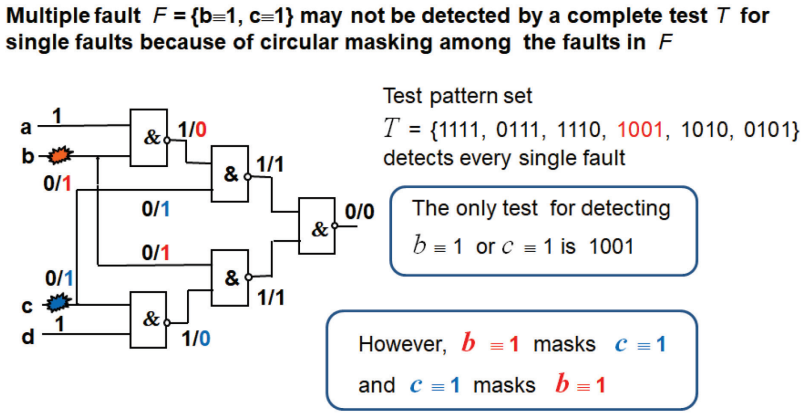


Figure 1.9: Multiple fault self-masking

The problem of multiple faults is illustrated in Figure 1.9.

The problem of multiple fault detection as a research issue dates back to 70s. The phenomenon of masking between faults [43] limits a test set derived for SSAFs from detecting MSAFs in complex digital circuits which may be redundant and may contain a large number of internal re-convergent fan-outs [61]. There are reports [62] which claim that a rather high coverage of MSAFs can be still achieved by SSAF test sets. But there are some critics as well on these results, mainly because of the chosen measure of fault

coverage [63]. A complete test set for SSAF, in general, is often incomplete for MSAF due to fault masking [64]. Moreover, under the assumption of SSAF, if the test set is incomplete, then an undetected SSAF may mask the presence of otherwise detectable fault. Therefore, multiple fault detection has become a necessity, and test generation for MSAF is of great interest for achieving high reliability of digital circuits.

In the past, a lot of effort has been made for developing methods to analyze MSAF detection capability in circuits for a given test [65]-[67]. In [66], during analysis, a frontier of SSAF is considered as an equivalent to a set of multiple faults currently simulated. In some works on MSAF testing structural constraints and masking relations of single faults are exploited to analyze the detectability of MSAF [68]-[69]. The problems of ATPG-based grading of self-checking properties and strong fault-secureness under the conditions of possible multiple faults are discussed in [70]-[71].

Different approaches have been used to reduce the complexity of test generation for MSAFs. In [72], an algorithm is developed to model a given MSAF as a single SAF. In [73], similarly, MSAFs are changed into equivalent SSAFs firstly, then an algorithm based on neural networks and chaotic searching is used to generate test vectors for SSAFs.

In [63], the target fault is considered as a single component of several MSAF, and the test is generated for that target fault regardless of the effects of other faults that might be present. Such a test is able to test all the multiple faults which contain the target fault as a component. A circuit decomposition based method is used in [74], where the tests are generated for MSAF in smaller blocks, and thereafter integrated into the complete test set for the whole circuit. The paper does not discuss the problem of fault masking in the case when the faults appear in different blocks. The problem of generating tests for SSAF that remain valid in the presence of undetected SSAF is discussed in [75].

Another approach to multiple test generation is using two-pattern tests (test pairs) [67], [76]-[78] to identify fault free lines. The method can deal with all combinations of multiple faults because each SSAF is considered independently. The method uses 16-value simulation on a pair of test patterns to detect MSAF. In [76], it was shown that a pair of input vectors is necessary for the valid test of a target fault under the multiple fault assumption, and an algorithm for generating a single sensitized path using a 7-valued calculus and a decision algorithm for finding a completely single sensitized path are presented. In [77], a method is proposed based on test pair analysis of the given test, and constructing additional pairs for undetected faults. The paper [78] presents a two phase method where first, the test pairs are found to detect the target SSAF independently of other faults, and thereafter, a sophisticated branch and bound procedure is used to complete the test set generation on the faults undetected during the first phase.

In [61]-[62], the test pair concept was used to generate test patterns for MSAF by targeting SSAF as test generation objectives. A different concept of test pairs for MSAF testing was proposed in [79], where the idea was not targeting the faults themselves as test objectives, but instead to interpret the test pairs as the means of identification of fault-free lines in the circuit.

However, in [80]-[81] it was shown that not always the test pairs can give a guarantee to avoid fault masking. In these papers, a concept of test groups was proposed. The main idea and goal of test groups was to verify the correctness of a selected subcircuit, instead of targeting the faults as test objectives, as it has been the traditional case. A complete test is generated as a set of test groups, and the pass of each test group identifies a fault-free part of the circuit regardless the effects of all possible faults which might be

present in other places of the circuit. The concept of fault diagnosis can be explained then as a method of extending step-by-step the fault-free core of the circuit.

The problems with test groups and with multiple fault diagnosis are twofold. First, the test groups may not always exist, which means that the multiple faults may not be detected, and the fault diagnosis will not be possible. Second, if a test group will not pass, it will be as well not possible to diagnose the fault location by extending step-by-step the fault-free core of the circuit.

In this Thesis (Chapter 5), a method is developed for fault diagnosis, which first, is not depending on how the diagnostic tests will be organized, and second, is immune to the cases if the responses to some test patterns will be disturbed due to fault masking.

2 TDF modeling and simulation

In this Chapter, a new advanced method of delay fault simulation is developed. First a new type of transition delay fault (TDF) model is introduced to improve the accuracy of measuring the delay fault coverage. Two novel algebras of delay fault reasoning and a new overall delay fault simulation procedure are presented for calculating the fault coverage with improved accuracy. Experimental data are presented, which demonstrate the advantages of the new methods compared to state-of-the-art.
The contribution of this chapter has been published in [82]-[86].

2.1 Modeling of TDF faults

To detect a Transition Delay Fault (TDF), two input patterns, $V = (V_1, V_2)$, must be applied, so that V_1 will initialize the circuit, while V_2 will activate the fault and propagate its effect to some primary output.

In the first phase of TDF simulation, SAF simulation will be carried out. For that, very fast parallel critical path tracing fault simulation can be used. As a result, all the nodes in the circuit, which were tested for SAF, are detected. For example, if the node x_i was tested for SAF $x_i \equiv 1$, then this node is a candidate for detecting the TDF "slow-to-fall" transition at this node, and if the node x_i was tested for SAF $x_i \equiv 0$, then the node is a candidate for detecting the TDF "slow-to-rise" transition at this node.

In conventional TDF testing it is needed that a test pair $V = (V_1, V_2)$ will assign to the node x_i different values, to initiate a transition, either $0 \to 1$ for detecting the TDF "slow-to rise", or $1 \to 0$ for detecting the TDF "slow-to-fall". These conditions are denoted as conditions to detect the TDF robustly. To determine, if the TDF fault $v \to v'$ is detected on the line x_i, where $v \in \{0,1\}$, and v' means inverted v, it is needed that the test pattern V_1 will assign $x_i = v$, the test pattern V_2 will assign $x_i = v'$, and the SAF fault $x_i = v$ is detected by V_2 at any primary output of the circuit.

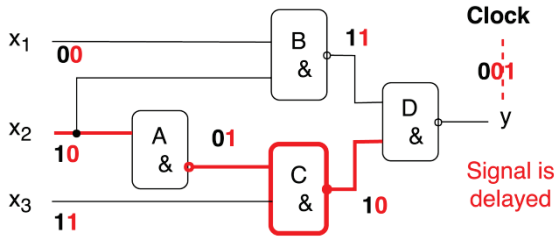


Figure 2.1: TDF testing along the signal path in the circuit

An example of such a delay test is shown in Figure 2.1, where $V_1 = 011(x_1, x_2, x_3)$, and $V_2 = 001(x_1, x_2, x_3)$. By SAF simulation of the test pattern V_2, it will be found that the following SAF will be detected:$x_2 \equiv 1$ at the input of gate A, $x_A \equiv 0$ at the input of gate C, and $x_C \equiv 1$ at the input of gate D. Hence, the test pair is testing the TDFs "slow-to-fall", "slow-to-rise", and "slow-to-fall", on the related inputs of gates A, C, and D, respectively.

In general case, e.g. when using the hazard-based detection conditions for the TDF $v \to v'$ on the node x_i, it is allowed to have a hazard or a pulse on the line x_i, under which x_i makes the transitions $v' \to v \to v'$ during the second pattern V_2 of the test. The fault is assumed to delay the $v \to v'$ transition of the pulse. Thus, in the presence of

fault, x_i is assumed to make the transition $v' \rightarrow v$, and stay at v. So, the fault can be detected by a test that creates the pulse on x_i and propagates the pulse under the second test pattern to an output. In other words, at the hazard-based conditions it is not necessary to create a transition on every line of the path to detect a TDF. It is also possible for lines on the path to have hazards. The TPDFs detected under hazard-based conditions correspond to a type of weak-non-robust test. This case is here called a non-robustly tested TDF.

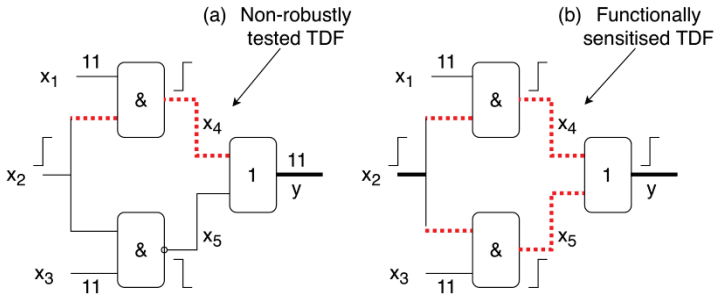


Figure 2.2: TDF is propagated along discontinuous paths

Consider an example in Figure 2.2, where a test pair $V = (V_1, V_2)$ is applied on the inputs with transition $0 \rightarrow 1$ on x_2. Assume that on both of the circuits the SAF faults at the outputs y are detected by V_2. In both circuits, the transition on the node x_2 is propagating further through two paths which re-converge at the node y.

In the case of Figure 2.2 (a), the fault effect on x_2 disappears, and does not reach the output y, because of different inversion parities on the re-converging paths. This is the reason why SAF $x_2 \equiv 0$ is not detected under V_2. Consequently, the TDF $0 \rightarrow 1$ on x_2 is not propagating to , and the TPDF on the path $x_2 - x_4 - y$ is not detected as well. However, the TDF $0 \rightarrow 1$ on x_2 can propagate non-robustly via x_4 to y, and hence detected non-robustly. As the result, the TPDF on the path $x_2 - x_4 - y$ can be detected non-robustly as well.

In Figure 2.2 (b), the TDF $0 \rightarrow 1$ on x_2 propagates to y, and is detected. However, the TDFs on x_4 and x_5 are not detected, because the SAFs $x_4 \equiv 0$ or $x_5 \equiv 0$ are not propagated under V_2. This means that there is no path in the circuit, where the TDFs are detected on all lines of the path, and consequently no TPDFs are detected in the circuit under test pair V. On the other hand, the TDFs at the nodes x_4 and x_5 are sensitized functionally, and can propagate to y as a multiple delay fault. This means that on both paths from x_2 to y, the TDFs are detected on all lines under alleviated conditions (under functional sensitization conditions on the lines x_4 and x_5), and hence, the TPDFs for the paths from x_2 to y are as well detected under functional sensitization conditions.

2.2 Extension of the class of TDF faults

To define the testing possibilities for TDFs and alleviate the constraints on detection of TDFs at the same time, the following types of TDFs were introduced: robustly tested TDF, i.e. the conventional case (denote it as R-type), non-robustly tested TDF (N-type), functionally sensitized TDFs, i.e. the multiple delay fault case (F-type), and a new type of non-robustly functionally sensitized TDFs (X-type).

These types of testing of TDFs are illustrated in Figure 2.3. For all the cases depicted, it is assumed that the SAF fault on the output of the gate is detected under the test vector V_2.

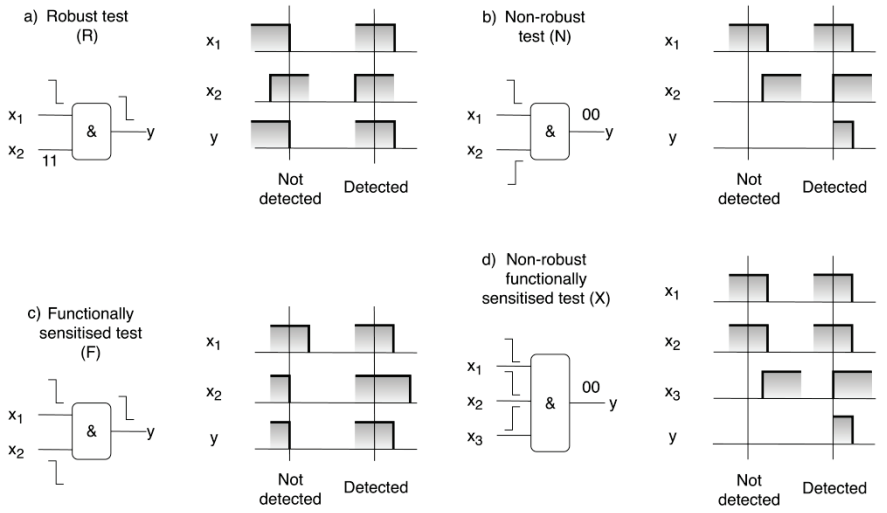


Figure 2.3: Four types of conditional detecting of TDFs

In all the cases in Figure 2.3 the TDF on the input x_1 is under test. The first diagram shows the waveforms of signals for the case where the delay fault on x_1 is missing or is not detected, and the second diagram shows the case where the delay fault is detected. In Figure 2.3a, the TDF on x_1 is tested robustly (conventional case or TDF of type R). Figure 2.3b shows the case of non-robust testing (the TDF of type N). The TDF on x_1 may not be detected if the transition $0 \rightarrow 1$ on x_2 is late. If this transition will be not delayed, or is delayed less than the transition on x_1, the TDF will be detected under the hazard-based conditions [38]. In Figure 2.3c, the TDF on x_1 is tested under functional sensitization conditions (the TDF of type F) [35]. This is the case when only multiple TDFs can be detected, i.e. the delay of the transition $1 \rightarrow 0$ on x_1 can be detected only if the same transition on x_2 is as well delayed. In the conventional meaning of TDF testing, the TDFs on the lines x_1 and x_2 can be regarded as untestable faults. Finally, in Figure 2.3d, the combination of the cases in Figure 2.3b and Figure 2.3c is considered (the TDF of type X). This is a new type of TDFs introduced in the paper. The multiple TDFs on x_1 and x_2 may not be detected if the transition $0 \rightarrow 1$on x_3 is late. If this transition will not be delayed, or is delayed less than the both transitions on x_1 and x_2, the TDF will be detected.

The TPDF model [49] for a path P requires that all the TDFs on the path P were detected. The detection of the TDF types like *N*, *F* and *X* allows alleviating the conditions under which the TDF will be considered as detected. However, the TDF detection conditions defined by *N*, *F* and *X* types are not guaranteed to happen. The goal of alleviation of the TDF detection conditions is to bridge the gap between the numbers of detectable TPDFs [49], and conventional PDFs, in a similar way as it was proposed in [38] by introducing the hazard-based TDF detection conditions (the type N of TDFs). Additionally, two more sensitizing conditions (F- and X-types of TDFs) were used in this chapter, to improve the exactness of TPDF coverage.

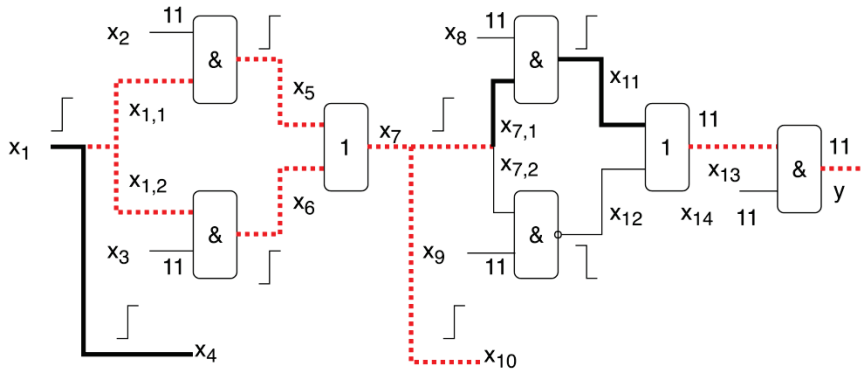


Figure 2.4: Detection of different types of TDFs

Table 2.1: Detected TDF faults

x_i	x_1	$x_{1,1}$	$x_{1,2}$	x_4	x_5	x_6	x_7	$x_{7,1}$	x_{10}	x_{11}	x_{13}	y
1	**R**	F	F	**R**	F	F	**R**	**R**	**R**	**R**	∅	∅
2	**R**	X	X	**R**	X	X	N	**R**	N	**R**	∅	∅

Example 2.1
Consider circuit in Figure 2.4 where the entire signal values of the given test pair are depicted. In Table 2-1, the detected TDF faults are shown for two cases:

1. The line x_{10} is tested via a subcircuit, not shown in the Figure for the R-type of TDF (TDF/R), similarly as in the traditional case of TDFs.
2. x_{10} is tested for TDF/N. Only the lines x_1, x_4, $x_{7,1}$, and x_{11} are tested in both cases robustly for TDF/R, as in the conventional case.

All other lines are not tested for TDFs in the traditional meaning of TDFs. However, if considering the 3 types of dynamic faults (types N, F, and X) introduced in the previous section, it can be seen that a lot of other lines can be tested for TDFs at different sensitizing conditions as well. If line x_7 is tested robustly (case 1), then the lines $x_{1,1}$, $x_{1,2}$, x_5, and x_6 are tested for TDF/F, otherwise for TDF/X (case 2). Explanation of that will be given in the next section.

2.3 7-valued algebra for TDF simulation

For simulating of TDFs and qualifying the types of the TDFs in accordance with the conditions they can be detected, two 7-valued algebras A1 and A2 were developed, as shown in Figure 2.5 and Figure 2.6 for the gates OR/NOR, and AND/NAND, respectively. The algebras are developed for two-input gates. If the number of inputs of the gate is bigger than two, then the gate has to be represented as an equivalent sequence of two-input gates. The algebras are defined for the set of values (symbols) $\{0,1,\varepsilon,h,N,F,X\}$, with interpretations explained in Figure 2.7, and by the binary calculation operators defined by Tables A1 and A2 in Figure 2.5, and Figure 2.6.

A1: Next signal propagation type

OR		Current signal propagation type						
		0	1	ε	h	N	F	X
Transition type	0	0	1	ε	h	N	F	X
	1	1	1	1	1	1	1	1
	ε	ε	1	F	N	X	F	X
	h	h	1	N	h	N	X	X

A2: Delay fault type

OR		Transition type	
		ε	h
Signal propagation type	0	Ø	Ø
	1	Ø	Ø
	ε	R	Ø
	h	Ø	R
	N	N	Ø
	F	F	Ø
	X	X	Ø

Figure 2.5: Two 7-valued algebras for the gates OR/NOR

The symbols 0, 1, ε, and h denote the signals and transitions on the inputs of the gate under analysis assigned by the test vector $V = (V_1, V_2)$. The symbols 0, 1 represent the steady signals 00 and 11, respectively, and the symbols ε and h represent the transitions 01 and 10, respectively. The symbols R, N, F and X denote the types of the conditions under which the TDFs can be detected. The symbol $\emptyset$ means that the TDF on the gate input under analysis is not tested. The symbol R is equivalent to the traditional case of testing TDFs.

A1: Next signal propagation type

AND		Current signal propagation type						
		0	1	ε	h	N	F	X
Transition type	0	0	0	0	0	0	0	0
	1	0	1	ε	h	N	F	X
	ε	0	ε	ε	N	N	X	X
	h	0	h	N	F	X	F	X

A2: Delay fault type

AND		Transition type	
		ε	h
Signal propagation type	0	Ø	Ø
	1	Ø	Ø
	ε	R	Ø
	h	Ø	R
	N	Ø	N
	F	Ø	F
	X	Ø	X

Figure 2.6: Two 7-valued algebras for the gates AND/NAND

Interpretation of symbolic values			
Transition types		Delay fault types	
0	00/No transition	F	Functionally sensitised delay
1	11/No transition	N	Non-robust testable delay
h	10/Falling transition	X	Non-robust functionally sensitised delay
ε	01/Rising transition	R	Robust testable delay

Figure 2.7: Interpretation of symbolic values of algebras A1 and A2

The fault types are engaged in the following dominance relation $R > N$, $R > F$, $R > F$, $N > X$, $F > X$. The higher is the ranking in this relation, the looser are the sensitization conditions for TDF detecting. Hence, it would be always advisable to choose the type of highest ranking if there is a possibility for a choice.

A sequential algebra for analysis of the input signals of the gate was introduced, because in the case of multi-input gates the signals impact is not independent of other signals when the type of the TDF propagation conditions is being calculated. In Figure 2.8, it is shown how the multiple input gate is represented as a sequence of 2-input gates for sequentially using Algebra A1 for calculation of the signals propagation type for the original multi-input gate. Then, by Algebra A2, the detectable TDF types for the inputs of the gate are calculated.

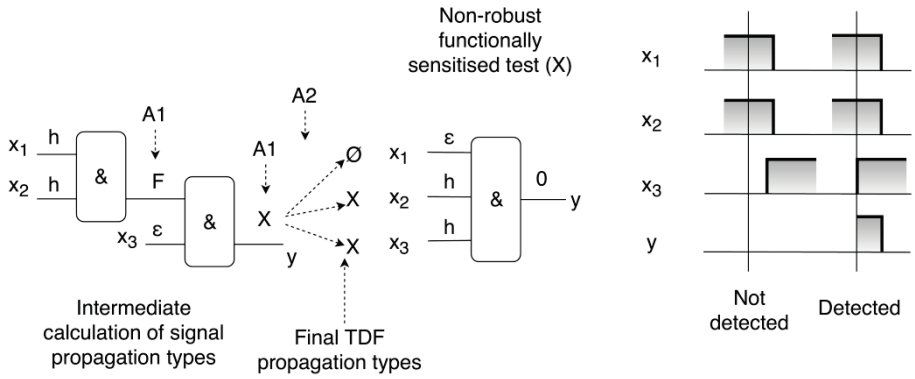


Figure 2.8: Using sequentially the Algebras A1 and A2 for TDF type calculation

The TDF simulation for the given test pair $V = (V_1, V_2)$, will be carried out according to the Procedure 1 in the following. Both, the SAF simulation and TDF simulation are carried out by back-tracing, starting from outputs towards inputs. One can assume that all the gates in the circuit are ranked in such a way that the outputs will have the highest rank, and the next rank can be assigned to a gate input line only if the gate already has a rank assigned.

2.4 Parallel critical path tracing procedure for robust TDF detection

In the first phase of TDF simulation the SAF faults in the circuit for the given set of test patterns are detected. For this purpose, the idea of parallel backward critical path tracing is used (first described in [87] and [27]), the main aspects of which are described in Section 1.2.

Consider a network of Fan-out-Free Regions (FFR) where each FFR is represented as a Boolean function $y = F(x_1, \ldots, x_i, x_j, \ldots, x_n) = F(X)$ and $X = (x_1, x_2, \ldots, x_n)$ is the input vector of the FFR. Such a network of five FFRs is represented in Figure 2.9.

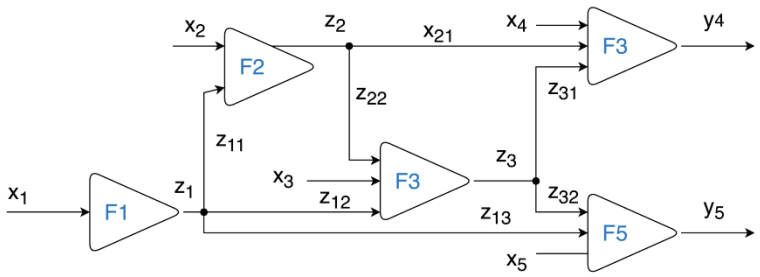


Figure 2.9: Combinational circuit as a network of five FFRs

Let us have the following notations:

- 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.

In [87] it was shown that if a SAF at y is propagated and detected by a test pattern at primary outputs then the fault at the fan-out stem x is as well detected if

$$\frac{\partial y}{\partial x} = y \oplus F\left(\left(x_1 \oplus \frac{\partial x_1}{\partial x}\right), \ldots, \left(x_i \oplus \frac{\partial x_i}{\partial x}\right), x_j, \ldots, x_n\right) = 1 \qquad (2\text{-}1)$$

The formula (2-1) taken in the vector form can be simplified as

$$\frac{\partial y}{\partial x} = y \oplus F\left(X' \oplus \frac{\partial X'}{\partial x}, X''\right) \qquad (2\text{-}2)$$

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, for calculating if the fault on z_2 can be detected on y_4 in Figure 2.9, it can be checked if

$$\frac{\partial y_4}{\partial z_2} = y \oplus F\left(X_4, z_{21} \oplus 1, z_{31} \oplus \frac{\partial z_3}{\partial z_2}\right) = y \oplus F\left(X_4, \overline{z_{21}} z_{31} \oplus \frac{\partial z_3}{\partial z_2}\right) = 1 \qquad (2\text{-}3)$$

The formula (2-2) can be used for calculating the influence of the fault (or erroneous signal) at the common fan-out stem *x* on the output *y* of the reconvergent fan-out region

by consecutive calculating of Boolean derivatives over related FFR chains starting from *x* up to *y*. For that purpose, for each fan-out stem involved in a reconvergent subnetwork, the corresponding formulas like (2-2) should be constructed. 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 subset of test patterns.

The whole process of fault simulation is carried out in two phases. In the first phase, parallel logic simulation is carried out to determine the logic values for all nodes in the circuit, which are needed to carry out the fault simulation. In the second phase the fault simulation using the formulas of type (2-2) is carried out for the simulated subset of test patterns.

To demonstrate the second phase of the fault simulation process, introduce the following notations for the formulas above which are used for calculating the Boolean derivatives:

- (x, y) – for $\partial y / \partial x$,
- $R_{xy}\big((x, x_1), \ldots, (x, x_k)\big)$ – for the general case (2-2), where $X' = (x_1, x_2, \ldots, x_k)$,
- Dx – vector which shows if the fault at the node *x* is detected or not detected at any circuit output.

Example 2.2
An example of a computational model of fault simulation for the circuit in Figure 2.9 is presented in Table 2.2.

Table 2.2: Example of critical path fault simulation for the circuit in

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 2.2 can be easily created and stored by the topological tracing of the circuit by algorithms developed in [87]. The algorithm has linear

complexity. However, the complexity of the computational model and the related fault simulation speed depends essentially on the structure of the circuit.

As the result of the described parallel critical path tracing procedure, the following two tables are created:

1. Fault table $FT = \|r_{i,j}\|, r_{i,j} \in \{0,1,\emptyset\}$ where i and j denote the numbers of simulated test patterns and nodes in the circuit, respectively; $r_{i,j} = 0(1)$ if the fault stuck-at $0(1)$ on the node j is detected by the pattern i, and $r_{i,j} = \emptyset$ if none of these faults is detected on the node j by the pattern i.
2. Test pattern table $TP = \|p_{i,j}\|$ where $\|p_{i,j}\| \in \{0,1\}$ is the signal value on the node j produced by the pattern i.

The data of tables FT and TP will be used for mapping SAF faults from FT to TDF candidates to be confirmed during the second phase of TDF simulation.

2.5 General procedure of TDF simulation with critical path tracing

In the following, a general procedure is proposed for calculating the different types of TDF faults using Algebras A1 and A2.

Procedure 2.1

1. First, using SAF simulator, a set of lines L will be determined for which the SAF fault is detected by the test pattern V_2.
2. For all the lines $l \in L$ where $V_1(l) \neq V_1(l)$, the TDF of type R will be fixed as detected (this determines the traditional TDF coverage).
3. For each gate with output line $\in L$, taken in the order of line ranking, the types of propagation of TDFs will be determined from the inputs of the gate to its output l. If an input line, not detected for TDF/R, will be assigned with a type TDF/D where $D \in \{N, F, X\}$, this line will be included into L. This step will be carried out by iterative using of Algebras A1 and A2.
4. In the process of simulating the given test sequence, current fault coverage will be updated by the simulation results for the current test pair in the following way:
    - for the lines where the TDF was detected the first time, the line will be marked as detected, and the given TDF type will be fixed;
    - if a line was already detected earlier for any TDF type, the type which has higher ranking in the dominance relation: $R > (F, N) > X$, will be chosen.

When analyzing a gate in Step 3 of Procedure 2.1, the following aspects will be taken into account.

1. If the input of the gate is already fixed as detected for TDF/R, the TDF type doesn't have to be recalculated.
2. If for the output line $l \in L$ of the gate a type $D \in \{N, F, X\}$ is assigned, then the transformation of the subcircuit will be applied in the opposite direction, compared to the case illustrated in Figure 2.8, and substitute the subcircuit by an equivalent multi-input gate. The problem is illustrated in Figure 2.10. Let the

output line of the NAND gate has the type N. First, the whole subcircuit will be substituted by the equivalent OR gate with inverted inputs x_1 and x_2. Then, value N is interpreted as the result of the analysis of the OR gate after using Algebra A1 for the inputs x_2 and x_3. After using iteratively Algebra A1 for the input x_1, the type X for the equivalent gate is captured. Now, using Algebra A2, the final types for x_1 and x_2 are calculated. Here it can easily be seen that the type N for the NAND gate has changed now to the weaker type X at the inputs. This is the same effect that could be noticed in the Example 2.1: the lines $x_{1,1}$, $x_{1,2}$, x_5, and x_6 in Figure 2.4 received the weaker types (with lower ranking) than the line x_7 (see also in Table 2.1).

3. For the lines with fan-out, the strongest type is assigned over the branches of the stem.

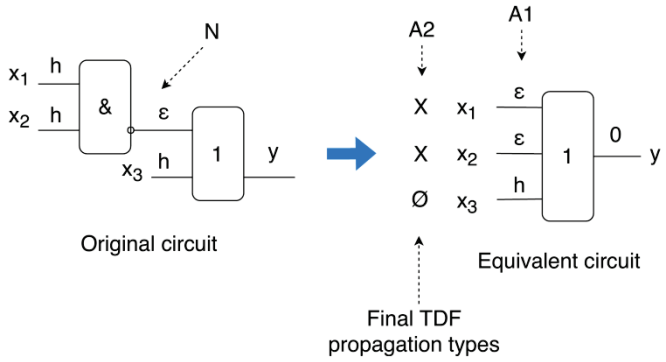


Figure 2.10: Backward propagation of TDF types

Example 2.3

Consider the procedure of using the algebras A1 and A2 for calculating the types of detected TDFs in Figure 2.11. For the output gate in Figure 2.2a, the inputs are following: $x_4 = \varepsilon$, and $x_5 = h$, which gives by A1 the type = N (non-robust). After what, it is found by A2 that there is no TDF tested for x_5 (the value $\emptyset$), but on the input x_4, the slow-to-rise TDF ($x_5 = \varepsilon$) will be tested non-robustly (the value N).

In Figure 2.11(a), the results of SAF simulation for the test vector V_2 are presented. The lines where SAF are detected are highlighted in bold blue color. The SAF simulation is carried out by critical back-tracing for both test-patterns as explained in Section 2.4. In general, the SAF back-tracing is carried out in parallel, however, only the second vector of the test pair $V = (V_1, V_2)$ needs delay fault reasoning. In Figure 2.11(b), the results of TDF analysis according to algebras A1 and A2 are presented. The analysis is carried out as well by back-tracing, starting always from the nodes where SAF is detected under V_2. The back-trace is continuing till the lines where no TDF is detected. For example, the lines x_{11}, x_6, and x_{72} do not need TDF analysis.

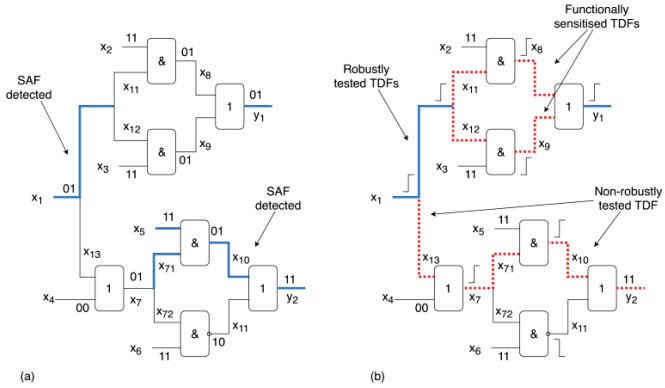


Figure 2.11: Example of conditional TDF simulation

The main practical goal of the described simulator is to improve the quality of the transition delay fault coverage calculation. In addition to the traditional robust delay coverage, the proposed simulator provides the possibility to get information about the conditional delay fault coverage for these faults which cannot be tested robustly.

2.6 Experimental results

Experimental research was carried out by TDF simulations for the ISCAS'85 circuits [88], where test pairs derived from the test set generated for SAF faults were used. The results are presented in Tables 2.3 and 2.4. The performances of the robust TDF simulator, created on the basis of SAF simulator, and the tool, developed for reasoning of the TDF types, are compared.

From Table 2.3 it can be seen that the TDF reasoning takes less time than the SAF simulation, and the more complex circuits are, the bigger is the difference. In average, based on the presented subset of experiments, the TDF analysis will consume less than 8% from the time needed for SAF simulation. On the other hand, as Table 2.4 shows, the speed of the SAF simulator proposed in the paper outperforms the performance of the fault simulators of the major CAD vendors. In Table 2.4 the speed of the SAF simulator is compared with FSIM [89], and with two state-of-the-art commercial fault simulators C1 and C2 for the benchmark circuits ISCAS'85 [88], ISCAS'89 [90] and ITC'99 [91]. Simulation times were calculated for the sets of random 10,000 patterns. Experiments were run on a 1.5 GHz Ultra SPARC IV+ workstation using SunOS 5.10.

Table 2.3: Experimental data of SAF and TDF simulation

Circuit	SAF simulation			TDF type calculation					
	Test length	Time ms	FC %	Test length	Time ms	Total	R	N	F
432	50	3	100	133	4.6	98.7	91.7	2.9	4.2
499	50	27	100	259	13.7	95.8	90.4	0.4	5.0
880	56	28	100	367	9.5	94.8	85.9	0.2	8.8
1355	100	48	100	320	21.0	95.7	91.3	0	4.4
1908	70	19	99.8	286	9.3	99.1	95.2	0	3.9
3540	178	167	100	680	29.3	100	99.6	0	0.4
5315	137	504	100	1159	72.5	100	99.6	0.1	0.3
6288	64	952	100	869	67.0	100	99.8	0	0.2

During the process of TDF simulation, a series of measuring was done to find out, how the fault coverage was evolving, as a function of the test length. The results for different TDF types for the circuit c432 are presented in Figure 2.12. In the simulation process, the dominance relations between different fault sensitization types were used represented as: $R > (F, N) > X$, where N and F have the same rank. If the same TDF is detected under different sensitization conditions, then the dominating type will be assigned to the TDF. This explains why the curves of N and F type of TDFs in the beginning of the fault simulation process are rising and later start to descend. In the end of the test, if not all TDFs can be tested robustly, they may be tested at-least in other N, F or X modes.

Table 2.4: Comparison of SAF simulation times

Circuit	Number of gates	SAF simulation time, s			
		Fsim	C1	C2	Proposed method
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

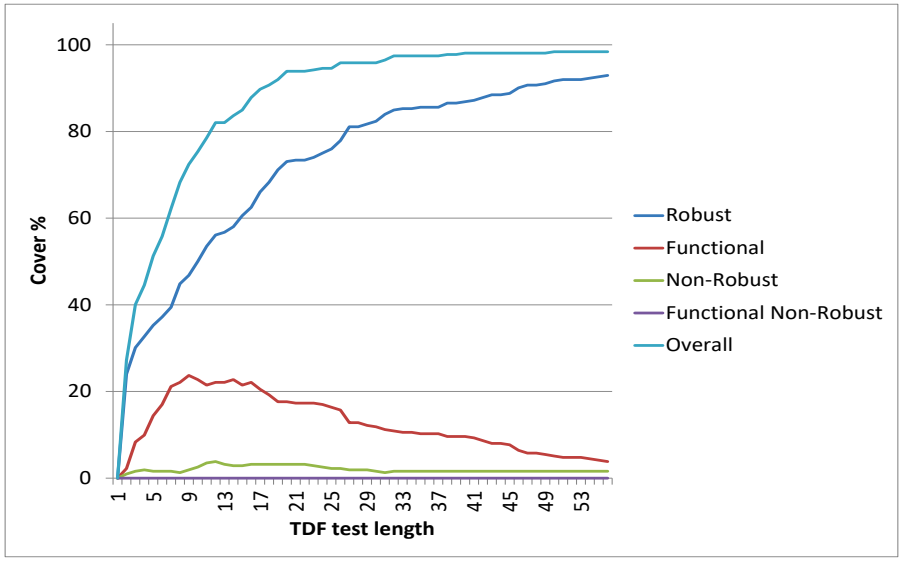


Figure 2.12: TDF covers as functions of the test length

The Figure 2.13 demonstrates how the structure of the sensitization types for TDFs changes in dependence of the initial SAF test quality. The TDF test was generated in a straightforward way from the initial SAF test. The lower was the SAF coverage of the

initial SAF test set, the higher was the share of conditional TDF coverage compared to the robust test coverage (the group of conditional TDFs includes the TDFs that are detected only under N, F or X type of sensitization).

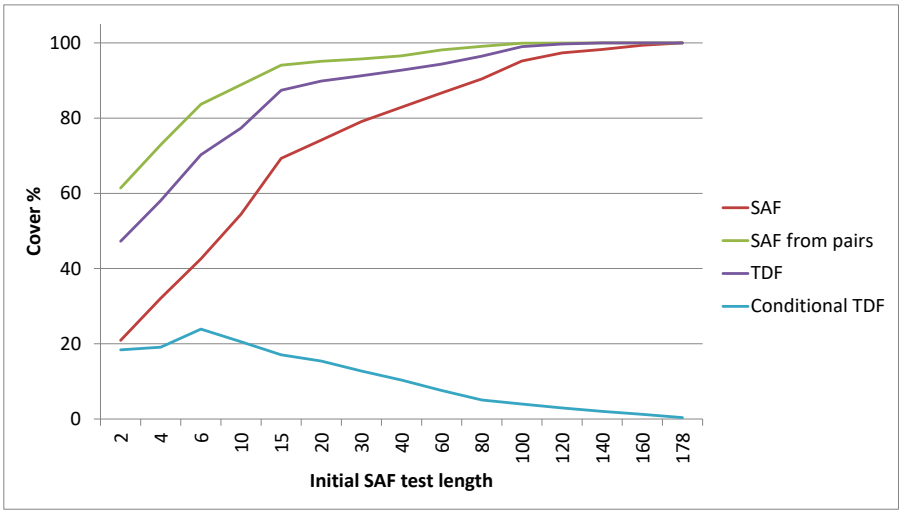


Figure 2.13: TDF covers as functions of the SAF cover basis

2.7 Conclusions

In this chapter, a new method for TDF simulation is proposed, which extends the set of conditions under which TDFs can be detected, compared to state-of-the-art [57] where only the hazard-based detection conditions have been considered.

The following new testing types were investigated for TDF detection, besides commonly used robust (traditional TDF model) and non-robust sensitization approaches: functionally sensitized (F-type), and as a new advanced propagation type – non-robust functional sensitization (X-type). Taking into account the two new sensitization conditions (F- and X-types) of TDFs allows to improve the accuracy of TPDF coverage, and to contribute to the advanced TPDF model developed in [49], [57].

The target for test generation should be to get all TDFs robustly tested. However, if it is not achievable for all TDFs, a part of them still may be tested under alleviated conditions. The proposed simulation procedure consists of two phases: traditional very fast parallel SAF simulation and additional analysis under which conditions the TDFs may be detected.

For robustly detectable TDF simulation, a fast pattern parallel exact critical path tracing method was developed, which surpasses in performance 2-4 times the state-of-the-art SAF simulators.

For simulating other types of conditionally detectable TDFs, introduced in the paper, a sequential 7-valued algebra was proposed. The fault simulation is as well based on the backtracing principle, which allows determining all the detected TDFs under different propagation conditions by a single run of the given test pair. The proposed method has a linear complexity.

3 Fault simulation in sequential circuits using DFT

In this chapter, a new method for fault simulation in sequential circuits is developed, where instead of the traditional fault-by-fault sequential simulation, a parallel fault simulation method is developed, which is based on reasoning concurrently of all the faults in the circuit, using the critical path tracing concept. To make this concept, originally developed for combinational circuits, applicable also for sequential ciruits, a simple update of a design is needed (design-for-testability), to virtually "cut" the global feedback loops by introducing a multiple input signature register (MISR). The one-cycle combinational critical path tracing algorithm is then generalized to multi-cycle sequential critical path tracing algorithm applicable for sequential circuits without global feedback loops, and consisting only local feedback loops caused by flip-flops. Experimental results are presented, which demonstrate dramatic speed-up of simulation, compared to traditional approaches

The contribution of this chapter has been published in [92].

3.1 Converting sequential fault simulation into combinational one using MISR

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 Sections 1.2 and 2.4, and which has been developed for using in 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 it will be shown, can be resolved if there will be a possibility to detect the fault in the first occasion when it has propagated up to any observable point without cycling in global loops.

There are two reasons why the same fault can be propagated to the same component of the circuit along different paths in different time frames:

1. The case of a global feedback loop which includes a component to which the same fault can have influence via different time frames;
2. The case of a sequential re-convergent fan-out, where the fault may propagate to the same component from the same fan-out stem via different time frames.

The first case 1 is illustrated in Figure 3.1. The circuit represented in Figure 3.1 consists of three registers and four combinational blocks connected by busses. Assume there is a fault Q in the sequential circuit on one of the outputs of the register R_2. Assume as well that the fault is propagating at the given test sequence to the primary output Y of the circuit by two successive test patterns. The first pattern propagates the fault in the clock cycle $t-1$ through combinational blocks F_3, F_4 and F_2 to the register R_3 (blue bold lines), and the second test pattern propagates the erroneous signal from the register R_3 in the next clock cycle t via block F_4 to the primary output Y (red bold lines).

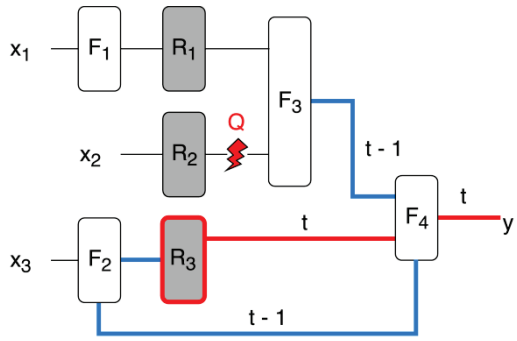


Figure 3.1: Critical path tracing of a fault in a sequential circuit (the case 1)

To better understand the mechanism of critical path tracing of faults in the sequential circuit, let us unroll the sequence of two test patterns into two time frames $t-1$ and t of the iterative logic array of the circuit presented in Figure 3.2.

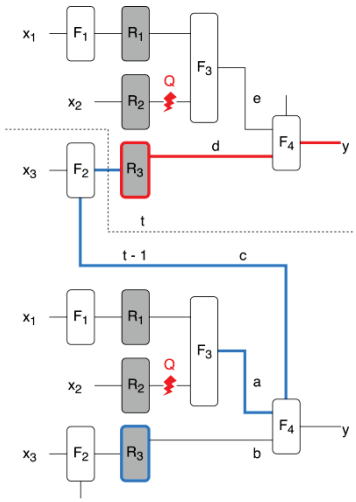


Figure 3.2: Critical path tracing of a fault in a sequential circuit over two successive time frames (the case 1)

In the iterative logic array in Figure 3.2, the single fault Q is propagating, and in such a way its impact is re-converging on the inputs of the block F_4 via two different paths: the path (Q, a, c, d) activated in the time frame $t-1$, and the path (Q, e) activated in time frame t.

Note that the critical path tracing of faults at the given test pattern is based on reasoning of the simulated correct signals in the circuit produced by the pattern.

Using only the correct signals makes it possible to determine in parallel all the faults which may propagate along the activated critical paths to the observable primary outputs. In Figure 3.2, there is activated a two cycle critical path along lines a, c (during cycle $t-1$), and d (during cycle t). The conditions of propagating the faults from a to c are determined by signals on the bus b at the time $t-1$, and the conditions of propagating the faults from d to the output Y are determined by signals on the bus e at the time t.

The critical path tracing of faults is carried out from the observable primary outputs towards the inputs. Starting from the output Y, the first step of the critical path tracing is to determine if the signals on the bus e are proper to propagate the faults from the line d through the block F_4 to the output Y. Since the fault Q under investigation should be considered in all time frames of the iterative array, the fault propagation conditions of the bus e are essentially depending also on the impact of the fault Q. However, this contradicts to the mechanism of critical path tracing of faults which must be carried out on the basis of using only the correct signals (also on the outputs of the register R_2).

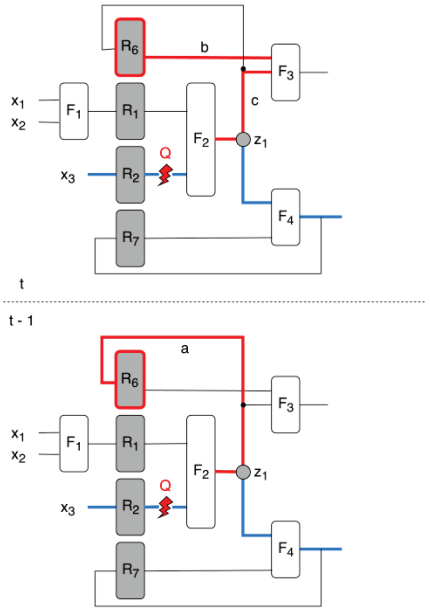


Figure 3.3: Critical path tracing of a fault in a sequential circuit over two successive time frames (the case 2)

The second case of a sequential re-converging fan-out is illustrated in Figure 3.3. In this case, the same fault is propagating to a converging point not along a global feedback, but rather along two branching paths activated at different time frames.

Assume, there is a fault Q in the sequential circuit on the output of the register R_1. Let us unroll the sequence of two test patterns into two time frames t and $t+1$ as shown in Figure 3.3. The first branch in Figure 3.3 consists of the path (Q, a, R_6) activated at time $t-1$, and of the path (R_6, b, F_3), activated at time t. The second re-converging branch is formed by the path (F_2, c, F_3), which is activated at time t. Both branches, propagating the impact of the same fault Q, are converging on the inputs of the block F_3 at the same clock cycle t.

For this case, it can be shown in the similar way as in case 1 that the critical path tracing, using only the correct signals in the circuit, is not possible. As an example, in Figure 3.3, in the time frame t, the faults on the line c cannot be back-traced, since the signal b at this time frame is not correct.

In such a way, the examples discussed on the basis of Figures 3.1, 3.2 and 3.3 demonstrate that in sequential circuits, each fault must be simulated separately, and the

classical fault independent critical path tracing of faults used in combinational circuits is not possible.

To convert the sequential case of fault interactions, illustrated in Figures 3.1, 3.2 and 3.3, into the combinational case, a possibility to process the faults immediately in the same time frame as they are sensitized, has to be introduced, to avoid processing the faults after propagation into the next time frame.

This can be done by introducing additional test points into the circuits for observing the problematic faulty signals before they will propagate to the next time frames. For that purpose, Multiple Input Signature Registers (MISR) can be used If a MISR is inserted and connected to the "problem causing" test points discussed above, the faults can be observed and detected always at the first occasion when they are sensitized. Once the fault is detected, its impact can be ignored in all the following test frames. In other words, these faults can be excluded from further simulation. Note, 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 the discussions above, two rules can be introduced to improve the observability of the sequential circuit.

RULE 1: Insert a MISR to all registers (and only to them), which are included into any global feedback. Inserting a MISR is equivalent to "virtual cutting" of the feedback loop (to give the possibility of ignoring the further propagation of detected faults on the related feedback loop).

RULE 2: Insert a MISR to all fan-out stems which have each at least one converging point, so that a faulty signal at this fan-out stem could propagate to this converging point at different time frames.

Consider a sequential circuit in Figure 3.4, which consists of 9 registers (latches) $R_1 - R_9$, and 8 combinational subcircuits $F_1 - F_8$. The circuit has 5 inputs and 2 outputs.

Let us analyze the circuit, and discuss the possibility or need of using the Rules 1 and 2 to make the circuit testable, so that the fault simulation can be processed according to the concept of critical path tracing in a similar way as in combinational circuits.

It is easy to see that two registers R_7 and R_8 in the circuit in Figures 3.3 are included into global feedback loops. According to Rule 1, the feedback loops must be virtually cut by introducing MISR for observing the registers R_7 and R_8. As the result, if any fault is propagating into these registers, the fault can be observed and detected by MISR. Being already detected, the faults can be excluded from the set of faults to be simulated by critical path tracing. Propagation of these faults into the next time frames is not needed.

Second, consider now the need of using Rule 2. There are two fan-out stems Z_1 and Z_2 in the circuit, which have two fan-out paths re-converging in the inputs of subcircuits F_3 and in F_7, respectively, in different clock cycles (different time frames). Hence, according to Rule 2, the nodes Z_1 and Z_2 must be monitored as well by MISR for observing the error signals in the fan-out stems immediately, to avoid self-masking of faults due to propagation of the same fault via different paths in different clock cycles to the same convergence point.

For example, a fault propagated to fan-out stem Z_1, can further propagate to the subcircuit F_3 in two different clock cycles – directly in clock cycle t, and via register R_6 in the next clock cycle $t + 1$. This makes impossible the fault back-tracing in subcircuit F_3 in the clock cycle $t + 1$, because the states of the lines in the subcircuit are corrupted do to the error captured in the register R_6 in the precious clock cycle t.

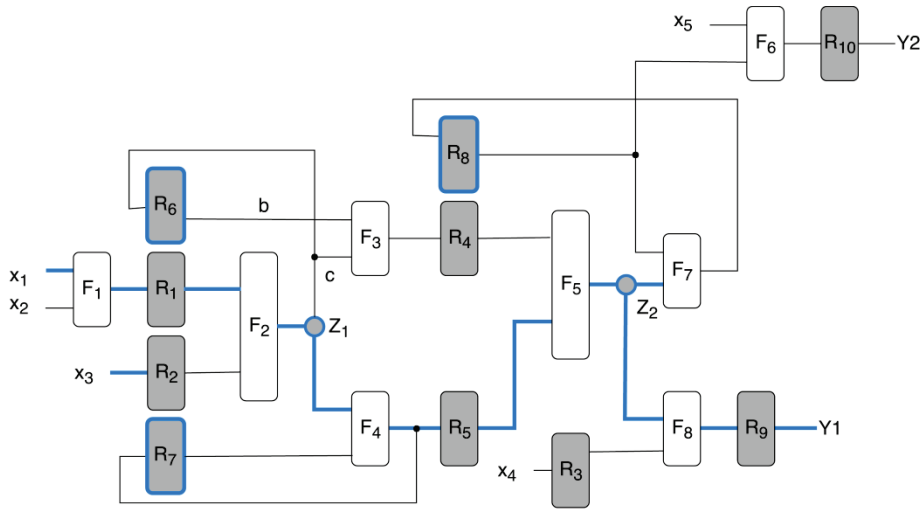


Figure 3.4: Sequential circuit with global feedback loops and sequential re-convergent fan-outs

In a similar way, a fault propagated to fan-out stem Z_2, can further propagate to the subcircuit F_7 in two different clock cycles – directly in clock cycle t, and via register R_8 in the next clock cycle $t + 1$.

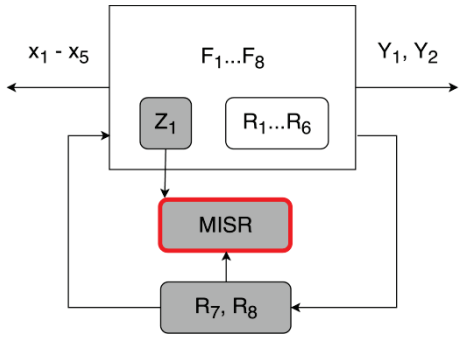


Figure 3.5: Sequential circuit with MISR

This makes impossible the fault back-tracing in subcircuit F_7 in the clock cycle $t + 1$, because the states of the lines in the subcircuit are corrupted do to the error captured in the register R_8 in the precious clock cycle t. In such a way, in the circuit in Figure 3.4, four test-points have been found - Registers R_7 and R_8, and the internal fanout points Z_1 and Z_2 (to be made observable by introducing MISR).

However, it is easy to see that the number of test points can be optimized, if the Rule 1 and Rule 2, will locate the same problematic test point. For example, in the present case, the test point Z_2 does not need observation, because one of the fan-out paths from it includes register R_8, which is already selected for observation by MISR. The modifications to be introduced into the circuit in Figure 3.4 are illustrated in Figure 3.5.

3.2 Compiling fault simulation model for the combinational part of the sequential circuit

For better focusing to the problem under discussion, and to skip the technical details of handling undefined states of registers which in general may not be initialized, the registers R_7 and R_8 with global feedback loops are assumed to be provided with RESET inputs RES_7 and RES_8, respectively.

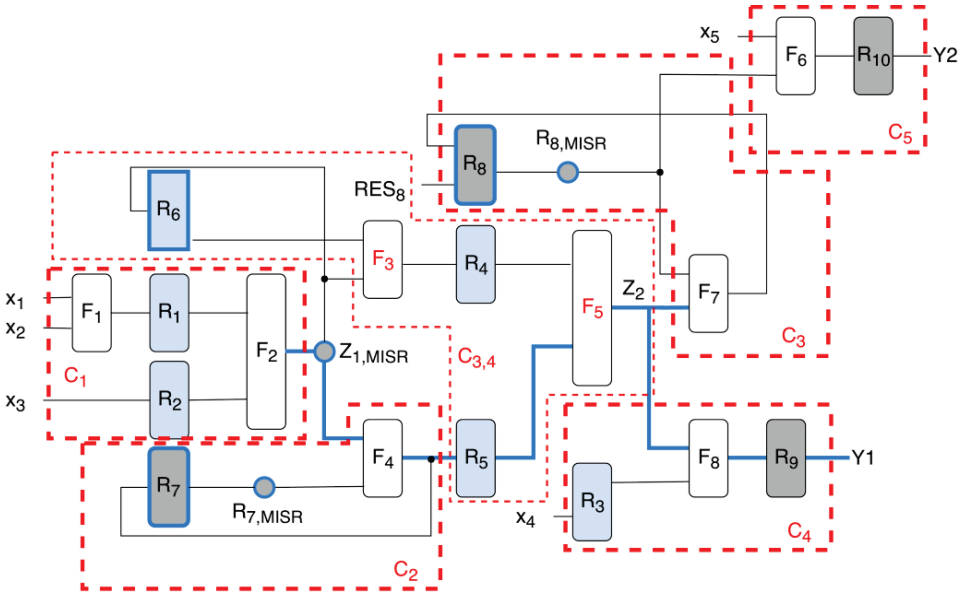


Figure 3.6: Circuit in Figure 3.5, partitioned into 5 equivalent combinational subcircuits

The nodes of the circuit which have been made observable by introducing MISR can be now considered as additional outputs of the circuit.

As the result of the described redesign for testability, which means introducing the MISR into the original circuit, the latter can be partitioned into 5 equivalent combinational subcircuits C_1, C_2, C_3, C_4, and C_5 as shown in Figure 3.6. The rest of the whole circuit denoted by $C_{3,4}$, should be considered as the joint subcircuit to be connected to both C_3, and C_4.

All the partitioned subcircuits have 5 outputs: primary outputs Y_1 and Y_2, and the test-points $R_{7,MISR}$, $R_{8,MISR}$, $Z_{1,MISR}$, which are connected to MISR. The inputs of the partitioned subcircuits may be either the 5 primary data inputs X_i, primary initialization inputs for registers R_7 and R_8 involved in the global feed-back loops, or the MISR variable $Z_{1,MISR}$.The partitioned subcircuits C_1, C_2, C_3, C_4, and C_5 do not have global feedbacks, and are forming a data-flow circuit, as shown in Figure 3.7, with functions (3-1), respectively.

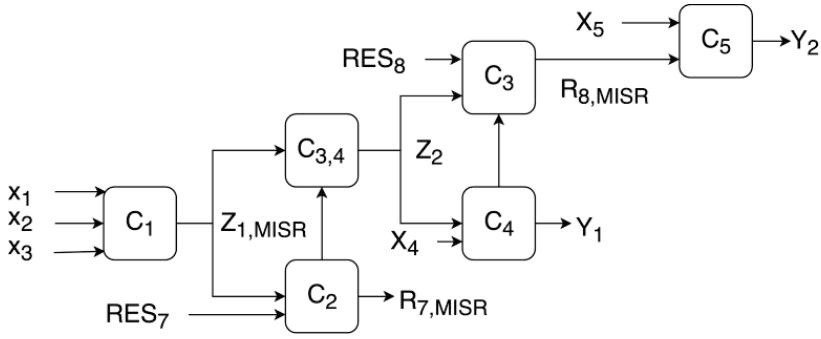


Figure 3.7: Circuit in Figure 3.5, partitioned into 5 equivalent combinational subcircuits

$$
\begin{aligned}
&Z^2_{1,MISR} = F(C_1) = \boldsymbol{F_2}(\boldsymbol{F_1}(X^1_1, X^1_2), X^1_3)\\
&R^3_{7,MISR} = F(C_2) = \boldsymbol{F_4}(Z^2_{1,MISR}, RES^1_7)\\
&R^5_{8,MISR} = F(C_3) = \boldsymbol{F_7}(RES^3_8, \boldsymbol{F_5}(\boldsymbol{F_3}(Z^2_{1,MISR}, Z^3_{1,MISR}), \boldsymbol{F_4}(Z^3_{1,MISR}, R^3_{7,MISR})))\\
&Y^5_1 = F(C_4) = \boldsymbol{F_8}(\boldsymbol{F_5}(\boldsymbol{F_3}(Z^2_{1,MISR}, Z^3_{1,MISR}), \boldsymbol{F_4}(Z^3_{1,MISR}, R^3_{7,MISR})), X^3_4)\\
&Y^6_2 = F(C_5) = \boldsymbol{F_6}(X^5_5, R^5_{8,MISR})
\end{aligned}
\qquad (3\text{-}1)
$$

In Figure 3.8, a simulation cycle of a test sequence with lengths of 5 clocks (5 input patterns) is shown; where by rectangles the 5 observation points are highlighted. In this simulation cycle, the processing of functions (3-1) can be observed. The upper indexes of the variables denote the numbers of clock cycles at which the argument variables are assigned with values and the function values are calculated).

All the functions (3-1) can be considered as representations of related "equivalent" combinational circuits, where the registers are substituted by wires which have the delay of one clock cycle. As the subcircuits C_1, C_2, C_3, C_4, and C_5 are combinational, a parallel critical path tracing of faults can be carried out in the circuit for the input patterns, where the values of variables should be taken from proper time frames. The tests can be applied to the circuit as pipelined sequences of test patterns. An example of two pipelined test sequences for the circuit presented in Figure 3.6 (or 3.7) is illustrated in Table 3.1.

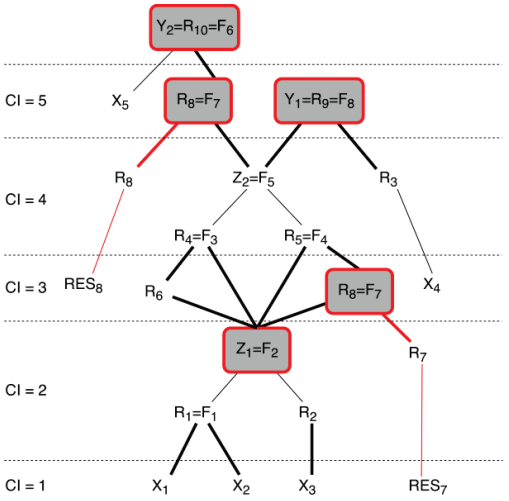


Figure 3.8: Simulation cycle of a single independent test

Table 3.1 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 (4-1). Because of the Rules 1 and 2 are satisfied in the modified circuit in Figure 3.5, 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 Figure 3.5 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 critical path fault tracing in parallel as in case of combinational circuits.

3.3 Experimental results

Experimental results are brought in Table 3.1, where the speed of SAF simulation in sequential circuits (where all the latches are fed into MISR) is compared to the parallel critical path tracing (PCPT) method described in Section 2.4 with different known fault simulators for combinational circuits: FSIM [89] 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.

In [93], a family of benchmark circuits were developed, which represent different architectures of a bio-impedance signal analyzer (a pipe-lined signal processor) with the same functionality. The feasibility of the proposed fault simulation method was investigated for calculating the fault coverage of the at-speed functional self-test developed for these processors.

Table 3.1: A test sequence for circuit in Figure 3.5

Cl	Input sequences		Output sequences		Simulated subcircuits	
	Test T_i	Test T_{i+1}	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				C_1	
2		X_1^2, X_2^2, X_3^2, RES_7^2	Z_1^2		C_1	C_1
3	RES_8^3, X_4^3		R_7^3, Z_1^3	Z_1^3	C_2	C_1
4		RES_8^4, X_4^4		R_7^4, Z_1^4	$C_{3,4}$	C_2
5	X_5^5		R_8^5, Y_1^5		C_3, C_4	$C_{3,4}$
6		X_5^6	Y_2^6	R_8^6, Y_1^6	C_5	C_3, C_4
7				Y_2^7		C_5

The results of fault simulation for the whole family of 8 processors (column 1) are presented in Table 3.3 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 PCPT shows the simulation time needed for the proposed method. The experiments showed that the gain achieved by using the proposed method is around 2-3 orders of magnitude. For this advantage the cost of added MISR has to be paid, which however is comparable to the cost of the scan-path. On the other hand, the proposed method has introduced dramatic speed-up in the test time, compared to the scan-path approach, and improved fault diagnosis.

Table 3.2: Comparison of PCPT with other fault simulation methods for circuits with full scan-path

Circut	**Number of gates**	**SAF simulation time, s**			
		Fsim	**C1**	**C2**	**PCPT**
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 the cases when the designs include logic BIST architectures which include already MISR facilities, the proposed approach of speeding up fault simulation does not add any additional area overhead cost.

Table 3.3: Comparison of the proposed method with single fault simulation in sequential circuits

Circuit	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

3.4 Conclusions

In this Chapter, a novel approach for fault simulation in sequential circuits is introduced that allows achieving 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 MISR for observing signals at selected test points.

The main novelties of the paper are as follows. Instead of full scan-path, usage of a MISR for monitoring the circuit in selected test points is proposed. As a consequence, instead of scan-path testing, at-speed functional test sequences can be used, which guarantee 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.

In the cases when the designs include logic BIST architectures which include already MISR facilities, the proposed approach of speeding up fault simulation does not add any additional area overhead cost.

4 Fault simulation in sequential circuit with parallel critical path tracing

In this Chapter, a new method for fault simulation in sequential circuits is proposed, where no updates in the design are needed, and the global feedback loops of the circuit are cut algorithmically. In the proposed approach, two methods are combined: fast parallel critical path tracing, and slow sequential fault-by-fault simulation. The set of all faults is partitioned algorithmically into two subsets, so that one of them can be simulated by the fast critical path tracing, and the rest of faults remain to be analyzed by the slower sequential fault-by-fault simulation. Experimental results show the speed-up of the new method compared to state-of-the-art.

The contribution of this chapter has been published in [94] and [95].

4.1 Converting sequential fault simulation into combinational critical path tracing

Consider a sequential circuit in Figure 4.1, partitioned into the combinational part, consisting of subcircuits A, B, and C with related fault sets S_A, S_B, S_C, and a sequential part consisting of flip-flops FF. Let $S = S_A \cup S_B \cup S_C$ be the full set of faults to be simulated in the circuit.

The faults in S_A will always propagate directly to the primary output OUT_A, and never to FFs. Hence, the faults in S_A can be handled in all time frames of the given test sequence independently and they can be simulated by the parallel critical path tracing (PCPT) approach. On the contrary, the effect of the faults in S_B can never be observed directly at OUT_A without propagating one or more times through the loop of the feedback FF, and hence, these faults are tending to be self-masked (similarly to the example in Figure 3.2 and Figure 3.3). Consequently, the faults S_B cannot be analyzed using PCPT, and they have to be processed by slow sequential fault simulation (SSFS) used traditionally in sequential circuits.

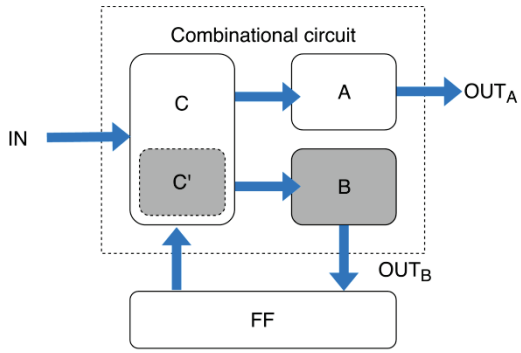


Figure 4.1: A sequential circuit partitioned according to fault types

The faults in S_C represent a special case. When a test sequence is applied to the circuit, the same faults in S_C may sometimes propagate directly to the primary output OUT_A, whereas in other times they may propagate first to OUT_B, and only then, after in the feedback loop, they may be observed at the primary output OUT_A. Let the faults in S_C, which propagate first to the pseudo-output OUT_B, form a subset $S_C' \subseteq S_C$. The described

ambivalence of the faults in S_C makes of them the key problem of the new fault simulation method.

The set of faults which propagate directly to the primary output OUT_A, without circulating in the feedback loop, can be represented as

$$S(A) = S_A \cup (S_C - S_{C'}) \tag{4-1}$$

These faults use for propagating to the observable output OUT_A a single time frame, and hence, can be simulated in the same way as in the combinational circuits using the very fast PCPT simulation approach. Note, the PCPT is carried out using the correct signals of the circuit, which is the prerequisite of the possibility of parallel analyzing concurrently all faults in the circuit.

The rest of faults

$$S(B) = S_B \cup S_{C'} = S - S(A) \tag{4-2}$$

need for propagating to the observable output OUT_A at least two (or more) time frames. This means that each fault, starting from the second time frame in its propagation trace up to OUT_A, will distort the states of signals in the circuit in all time frames in its own way, making the concurrent analysis of different faults not possible. Hence, all the faults in $S(B)$ have to be simulated separately by conventional methods used for sequential circuits, which is a very slow procedure.

Since the conventional fault simulators for sequential circuits are targeting by this slow approach all faults S in the circuit, it would be promising to extract from S the subset of faults $S(A)$ which can be simulated using fast PCPT. In this way, it would be possible to speed up dramatically the full fault simulation procedure.

Here the following simulation based method is proposed, for classification of all faults into two subsets $S(A)$ and $S(B)$ where $S(A) \cap S(B) = \emptyset$.

The difficulty is related to the fault set $S_{C'}$, because the content of $S_{C'}$ is strongly dependent on the test sequence. In other words, it is not possible to predefine $S_{C'}$ before fault simulation, and the content of $S_{C'}$ can be defined only "on-line" during the process of fault simulation. This additional payload may slow down the full procedure.

The method, illustrated in Figure 4.2, is based on applying critical path tracing for the patterns in sequence, one by one, separately for the outputs OUT_A and OUT_B. Latter method results in two targets being hit at once: first, part of faults $S(A)$, will be immediately found by using critical path tracing; and second, the subset of faults $S(B)$ which will require a separate slow fault simulation, will be extracted.

Process starts from the first pattern of the test sequence, and calculates the sets S_A^1 and S_B^1 of faults detected on the outputs OUT_A and OUT_B, respectively. The faults of S_A^1 can be included immediately into $S(A)$ of detected faults by this first pattern, $S(A_1)$=S_A^1, whereas the faults S_B^1 are included into $S(B_1)$.

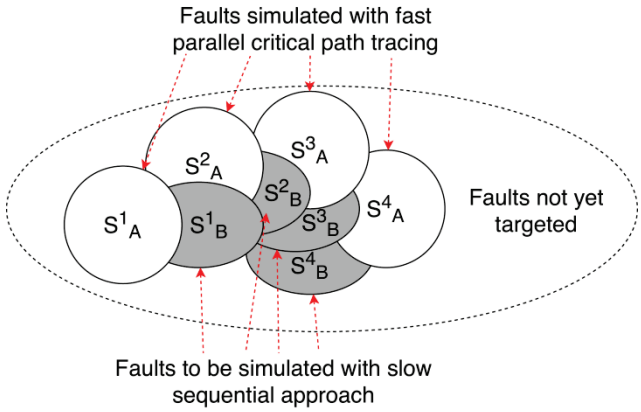


Figure 4.2: Simulation based fault classification

Next, for the second pattern of the test sequence, the sets of detected faults S_A^2 and S_B^2 are found. The sets $S(A)$ and $S(B)$ can be then updated in the following way:

$$S(A) = S(A_2) = S(A_1) \cup \left(S_A^2 - S(B_1)\right) \tag{4-3}$$

$$S(B) = S(B_2) = S(B_1) \cup \left(S_B^2 - S(A_1)\right) \tag{4-4}$$

Following the example in Figure 4.2, the state of the fault simulation can be expressed in the general case after the k^{th} pattern of the test sequence:

$$S(A) = S(A_k) = S(A_{k-1}) \cup \left(S_A^k - S(B_{k-1})\right) \tag{4-5}$$

$$S(B) = S(B_k) = S(B_{k-1}) \cup \left(S_B^k - S(A_k)\right) \tag{4-6}$$

The procedure ends if the k^{th} pattern will be the last pattern of the test sequence. $S(A)$ will represent the set of faults simulated and detected very fast by the critical path tracing method, and $S(B)$ will represent the faults which need additional fault simulation by any conventional fault simulator for sequential circuits.

4.2 Combining parallel critical path tracing with sequential fault simulation

In the previous Section, a test for a sequential circuit was considered as a single test sequence where all test patterns consisting of primary input patterns and pseudo-input patterns (the output values of flip-flops), are strongly depending on each other due to the feedback loop. This fact allowed us to exploit the power of critical path tracing only for single patterns. In Section 2.4 the method of parallel critical path tracing concurrently for many patterns, developed for combinational circuits, was described. This parallelism is possible due to the independency of test patterns in case of combinational circuits.

In the following, a method for parallel critical path tracing was developed also for sequential circuits, which exploits the independences between the test segments in the full test sequence. Assuming that each segment will consist of the initialization of the

flip-flops, and the subsequent patterns will serve for fault sensitization and fault propagation to the primary output OUT_A.

Consider a test sequence consisting of k test segments, where each i^{th} test segment consists of m_i test patterns $TS_i = (T_{i,1}, T_{i,2}, \ldots, T_{m_i})$. Since all test segments are independent, then also the first patterns $(T_{1,1}, T_{2,1}, \ldots, T_{k,1})$ in all k segments, and in a similar way, the second patterns $(T_{1,1}, T_{2,1}, \ldots, T_{k,2})$ in all k segments, etc., can be considered together as a set of m packages $\{TPP_i\}$ of independent test patterns. The lengths of the packages, in general case, may be different, and the number of packages m is equal to the number of the patterns in the longest test sequence.

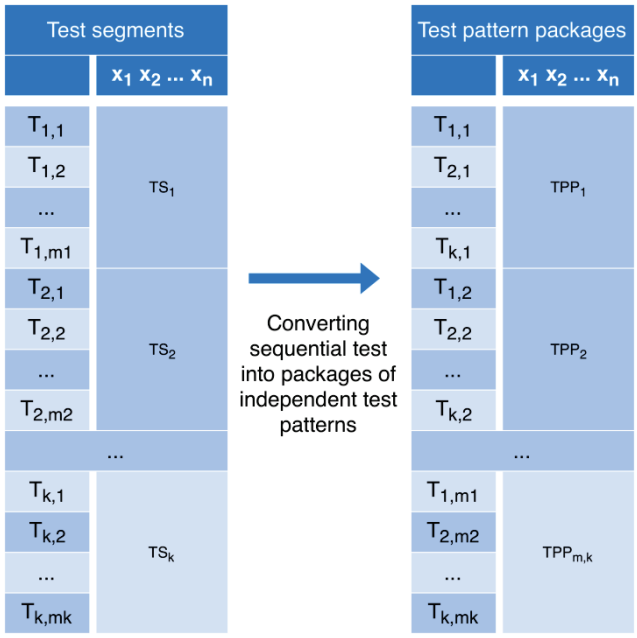


Figure 4.3: Converting the set of test segments into the set of independent test pattern packages

An example of converting the initial test sequence into a set of packages of independent test patterns is depicted in Figure 4.3.

For each package $TPP_i = (T_{1,j}, T_2, j, \ldots, T_{k,j}), j = 1,2,\ldots,m,$ of test patterns, parallel critical path tracing can be applied concurrently for all patterns in the package. As the result of this procedure, for each test pattern $T_{i,j} \in TPP_j$, the fault sets $S_A^{i,j}$ and $S_B^{i,j}$ will be calculated, and the detected fault sets (fault covers) will be calculated in the following way.

$$S_A^j = S_A^{1,j} \cup S_A^{2,j} \cup \ldots \cup S_A^{k,j} \tag{4-7}$$

$$S_B^j = S_B^{1,j} \cup S_B^{2,j} \cup \ldots \cup S_B^{k,j} \tag{4-8}$$

An example of a fault table created by parallel critical path tracing for the given set of k packages of independent test patterns is depicted in Figure 4.4.

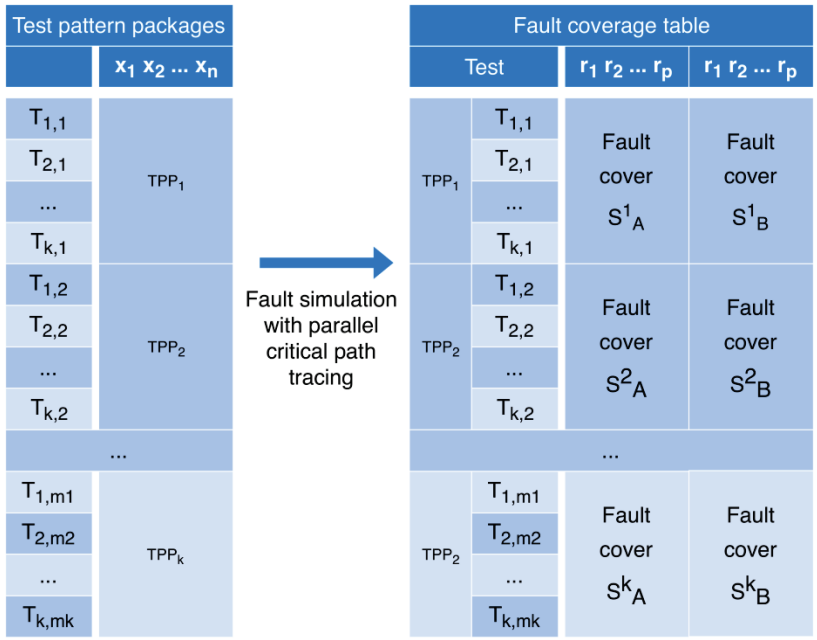


Figure 4.4: Results of parallel critical path tracing of sequential test

Based on the fault sets S_A^j and S_B^j calculated by critical path tracing, the sets of faults $S(A_k)$ and $S(B_k)$ can be calculated similarly to the formulas (4-5) and (4-6), respectively.

The full procedure of the method of combining parallel critical path tracing with sequential faults simulation for sequential circuits can be presented as follows.

Algorithm 4.1

1: Convert the set of test segments into test packages of independent test patterns (see Figure 4.3)

2: **for** each test package TPP_j

3: Apply parallel critical path tracing to calculate the fault sets S_A^j and S_B^j

4: **end for**

5: **for** $k = 1,2,\ldots,m$ (m is the number of test packages, $S(A_0) = S(B_0) = \emptyset$)

6: $S(A_k) = S(A_{k-1}) \cup \left(S^k{}_C - S(B_{k-1})\right)$

7: $S(B_k) = S(B_{k-1}) \cup \left(S^k{}_B - S(A_k)\right)$

8: **end for**

9: **return** $S(A) = S(A_m), S(B) = S(B_m)$,

The set of $S(A)$ represents the faults detectable by the given test, calculated by parallel critical path tracing (PCPT). The set of $S(B)$ represents the faults whose detectability is not possible to determine by PCPT. The faults in $S(B)$ have to be simulated by conventional fault simulation methods used for sequential circuits.

Let us calculate now the characteristics of the timing analysis and of possible speed-up of the proposed method. Let us use the following notations:

- n – the number of faults in the circuit;
- n_B – the number of faults in $S(B)$ to be simulated by slow sequential fault simulator;

- t_{anal} – the total time needed for critical path tracing and fault analysis consisting of two parts

$$t_{anal} = t_{CP} + t_{CL}$$

where

- t_{CP} – the time needed for critical path tracing, and
- t_{CL} – the time needed for fault classification.
- t_{OLD} – the time needed for fault simulation of a fault in a sequential circuit by a conventional fault simulator;
- t_{seq} – the average time of sequential simulation of a fault

$$t_{seq} = t_{OLD}/n$$

The total time needed for the proposed method which combines the critical path tracing with conventional sequential fault simulation can be calculated as

$$t_{NEW} = t_{anal} + (t_{OLD} \times n_B) = t_{CP} + t_{CL} + (t_{OLD} \times n_B) \quad (4\text{-}9)$$

The speed-up of using the proposed method compared to the sequential approach can be characterized as

$$\frac{t_{OLD}}{t_{NEW}} = \frac{t_{seq} \times n}{t_{anal} + (t_{seq} \times n_B)}$$

Denote $p = t_{anal}/(t_{seq} \times n_B)$.
Since

$$t_{anal} \ll t_{seq} \times n_B$$

it would be reasonable to calculate the higher limit of speed-up when $p \to 0$:

$$\lim_{p \to 0} \frac{t_{OLD}}{t_{NEW}} = \lim_{p \to 0} \frac{n \times t_{seq}}{n_B \times t_{seq}} = \frac{n}{n_B} \quad (4\text{-}10)$$

Equation (4-10) demonstrates, that the speed-up of the proposed method is substantially depending on the number of faults n_B in $S(B)$, which cannot be simulated by critical path tracing. Note, the content of $S(B)$, and the speed-up effect of the proposed method depends also on the given test sequence to be fault simulated.

4.3 Experimental data

The goal of the experimental research was to measure and demonstrate the speed-up of the proposed new method for fault simulation in sequential circuits compared to the conventional fault simulation approach. The comparison was carried out for a representative selection of 27 circuits with different complexities of two benchmark families ISCAS'89 [59] (16 circuits) and ITC'99 [60] (11 circuits). The numbers of possible SAF faults in the circuits ranged from 614 up to 129422

Experiments were run on Intel i7-6700 4 cores 3.4 GHz, 32 GB RAM 2133 MHz, Windows 10, 64-bit.

Table 4.1: Characteristics of the benchmark circuits used in experiments

Circuit	Inputs, #	Outputs, #	Flip-Flops, #	Outputs/Flip-Flops	Gates, #	Faults, #
s349	9	11	15	0,7	161	614
s386	7	7	6	1,2	159	744
s510	19	7	6	1,2	211	900
s526	3	6	21	0,3	193	984
s641	35	24	19	1,3	379	1164
s713	35	23	19	1,2	393	1266
s953	16	23	29	0,8	395	1720
s1423	17	5	74	0,1	657	2610
s1494	8	19	6	3,2	647	2864
s5378	5	49	179	0,3	2779	9122
s9234	19	22	228	0,1	5597	16756
s13207	31	121	669	0,2	7951	24882
s15850	14	87	597	0,1	9772	29682
s35932	35	320	1728	0,2	16065	65248
s38417	28	106	1636	0,1	22179	69662
s38584	12	278	1452	0,2	19253	72346
b04	11	8	66	0,1	652	2836
b05	1	36	34	1,1	927	3952
b07	1	8	49	0,2	383	1712
b08	9	4	21	0,2	149	706
b10	11	6	17	0,4	172	806
b11	7	6	31	0,2	726	2894
b12	5	6	121	0,0	944	4426
b13	10	10	53	0,2	289	1350
b14	32	54	245	0,2	9767	38982
b15	36	70	449	0,2	8367	36496
b17	37	97	1415	0,1	30777	129422

Table 4.1 presents the main characteristic data of the circuits used in experiments. The following values were observed: the number of inputs (#), the number of outputs (#), the number of flip-flops (FF#), the ratio between the numbers of observable primary outputs and not directly observable flip-flops Out#/FF#, which characterizes the sequential complexity of the circuit, the number of gates (#) and the number of possible stuck-at-faults (#).

Table 4.2 presents data of the experimental research, where the benchmark circuits are ordered according to their increasing complexity (the numbers of faults).

To each benchmark circuit a sequential test was applied, consisting of 32 independent test segments, where each segment was created using a sequence of 50 random input patterns. The initial assumption was that before each test sequence the circuit flip-flops were initialized (reset). The total length of the test sequence was 1600 test patterns.

In columns 4-6, the test sequences are characterized by the fault coverage: $s(A)$ and $s(B)$ mean the percentages of faults in the sets $S(A)$ and $S(B)$, respectively, in relation to the full set of faults given in column 2. The sum $s(A) + s(B)$ in column 6 characterizes the sets of faults which propagate to the primary outputs and flip-flops, respectively. Note, the faults in $S(A)$ are detected directly by the critical path tracing, but the faults in $S(B)$ are those which need additional sequential fault simulation to determine if they propagate as well to the primary outputs.

Table 4.2: Experimental results of comparing the proposed method vs. conventional fault simulation methods for sequential circuits

No	Benchmark circuits s - Iscas'89 b - ITC'99	Complexity of circuits (# faults)	Fault coverage of the simulated test sequence (%)			Time costs for critical path tracing of faults (s)		Total time costs for fault simulation (s)			Results	
								t_{NEW}	t_{OLD}		Gain t_{OLD_NFD}/t_{NEW}	Higher bound
			s(A)	s(B)	S(A)+S(B)	t_{CP}	t_{CL}		t_{OLD_FD}	t_{OLD_NFD}		
1	2	3	4	5	6	7	8	9	10	11	12	13
1	s349	614	8,3	88,1	96,4	0,10	0,01	0,5	0,03	0,44	0,89	1,13
2	b08	706	2,3	96,0	98,3	0,11	0,01	0,6	0,05	0,51	0,84	1,04
3	s386	744	36,0	36,0	72,0	0,12	0,01	0,3	0,10	0,49	1,61	2,78
4	b10	806	3,0	89,5	92,4	0,10	0,01	0,7	0,08	0,65	0,93	1,12
5	s510	900	28,2	65,1	93,3	0,11	0,02	0,6	0,12	0,73	1,21	1,54
6	s526	984	3,3	32,3	35,6	0,10	0,01	0,4	0,25	0,80	2,14	3,09
7	s641	1164	50,0	37,2	87,2	0,12	0,03	0,7	0,16	1,56	2,15	2,69
8	s713	1266	43,8	37,6	81,4	0,12	0,03	0,8	0,23	1,75	2,17	2,66
9	b13	1350	2,5	62,0	64,5	0,12	0,02	1,2	0,41	1,72	1,43	1,61
10	b07	1712	1,1	79,8	80,8	0,14	0,05	2,3	0,47	2,70	1,15	1,25
11	s953	1720	8,0	86,3	94,4	0,16	0,03	2,5	0,31	2,73	1,07	1,16
12	s1423	2610	2,3	68,5	70,8	0,16	0,05	4,6	1,04	6,38	1,39	1,46
13	b04	2836	1,1	89,9	91,0	0,18	0,15	7,4	0,57	7,84	1,06	1,11
14	s1494	2864	37,9	20,2	58,1	0,15	0,07	1,5	1,38	6,28	4,22	4,95
15	b11	2894	0,8	67,3	68,2	0,16	0,04	5,3	1,58	7,60	1,43	1,48
16	b05	3952	2,2	32,2	34,4	0,21	0,16	4,5	4,22	12,99	2,86	3,11
17	b12	4426	0,5	38,4	38,9	0,20	0,07	6,3	4,25	15,83	2,50	2,61
18	s5378	9122	45,7	26,2	71,9	0,36	0,24	22,9	11,00	85,21	3,72	3,82
19	s9234	16756	2,1	30,1	32,2	0,58	0,36	91,1	105,00	300,03	3,29	3,33
20	s13207	24882	6,3	45,7	52,0	0,84	0,51	306,2	139,00	667,20	2,18	2,19
21	s15850	29682	5,1	32,5	37,6	1,17	0,60	309,6	250,00	946,45	3,06	3,08
22	b15	36496	0,6	49,7	50,3	5,35	0,65	521,2	228,60	1036,21	1,99	2,01
23	b14	38982	0,5	74,6	75,1	3,48	0,73	919,3	258,80	1226,38	1,33	1,34
24	s35932	65248	8,8	75,6	84,4	1,51	0,95	2827,0	367,00	3734,21	1,32	1,32
25	s38417	69662	1,9	50,7	52,6	2,44	1,35	2516,8	1189,00	4956,59	1,97	1,97
26	s38584	72346	8,0	69,9	77,9	2,27	1,48	3255,9	704,00	4650,52	1,43	1,43
27	b17	129422	0,3	25,0	25,3	15,42	1,96	3226,2	2856,20	12857,76	3,99	4,01
Average						**1.33**	**0.35**	**519.9**	**226.81**	**1130.80**	**2.0**	**2,2**

All simulation time costs are given in seconds. The time costs t_{CP} and t_{CL} in columns 7 and 8 characterize the time needed for critical path tracing and the time needed for fault classification, respectively. Hence, the time $t_{CP} + t_{CL}$ represent the full cost of the parallel fault simulation of faults by critical path tracing. The total time cost of fault simulation by the new method proposed in the paper, where the parallel critical path tracing and the traditional sequential fault simulation are combined, is depicted in column 9. Columns 10 and 11 present the time costs of the baseline methods – the traditional sequential fault simulation. Here, two approaches are considered – simulation with fault dropping [3] (t_{OLD-FD}), and simulation with non-fault dropping ($t_{OLD-NFD}$). In column 12 the advantage (the gains in time cost) of the proposed method is presented, and in column 13, for comparison, the higher bounds of the possible gains are depicted.

Note the target of this research was not to generate test patterns with highest fault coverage, rather, the goal of experiments was to investigate the speed-up of fault simulation, compared with the baseline, for any test sequences generated and supposed to be the objective of fault simulation. In the current case, randomly generated test sequences were selected with equal test length for all circuits.

4.4 Discussion of the experimental results

The test sequences were analyzed twice. First, by the proposed method of parallel critical path tracing, the sets of faults $\{S_A^k\}$ which propagated to the primary outputs OUT_A, and the sets of faults $\{S_B^k\}$ which propagated to the pseudo-outputs OUT_B , were calculated. The time cost of this procedure was measured as t_{CP}. Then the sets of faults $S(B) \subset \{S_A^k\}$ and also the sets of faults $S(B) \subset \{S_B^k\}$, detected by the proposed method, and which have to be simulated by any conventional simulator of sequential circuits, were classified. The time cost of fault classification procedure was measured as t_{CL}.

The experimental results of the gain in speed-up of the new method, compared to the baseline, in column 12, and the higher bounds for the gain, calculated by the formula (6), are illustrated in Figure 4.5. The new method outperforms the baseline method in a broad range of 1.04 -5 times (in average 2 times), except of only very small circuits (less than 200 gates).

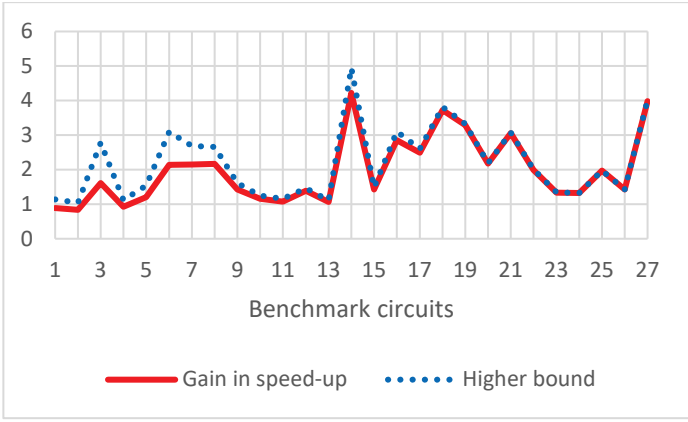


Figure 4.5: Comparison of the speed-up of simulation with higher bound

On the other hand, it can be observed that the experimental results and the theoretically calculated bounds are very close in case of small circuits. In case of larger circuits, they nearly match. This gives an excellent possibility to predict by a simple calculation of the formula (6) the expected speed-up of fault simulation by the new proposed method.

Figure 4.6 illustrates the dependence of the gain in speed-up of simulation on the complexity of circuits (the number of faults). The cumulative gain and the cumulative number of faults over the full set of benchmark circuits used in the experiments were compared. The graphics show that the gain is increasing nearly linearly over the full set of circuits ranked in order of complexity. It is evident that the linearity also takes place in the second part of the circuits where the complexity starts to increase very rapidly.

This finding allows claiming that the proposed method is well scalable, since the method starts to work more efficiently in case when the circuit complexity will increase.

The cumulating functions were used in the graphics in Figure 4.6 to smooth the anomalies of the gain curve in Figure 4.5, and the nonlinearity of spreading the number of faults on the X-axis.

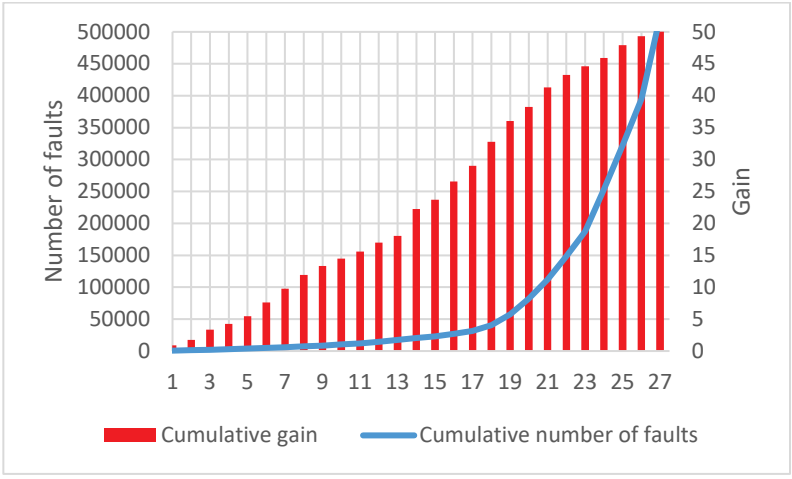


Figure 4.6: Dependence of speed-up of simulation on the complexity of circuits

This anomaly can be explained by the two charts in Figure 4.7, which represent the values of feedback level and the gain in speed-up of simulation for the full range of benchmark circuits.

To characterize the feedback level, the values of $s(B)$ in Table 4.2 were used, representing the percentage of faults that do not propagate to the primary output, but rather start propagating via feedback loops to the subsequent clock cycles. These faults are representing the bottleneck of the proposed method, because they cannot be simulated by fast critical path tracing.

The Figure 4.7 demonstrates that for all circuits where $s(B)$ is high, the gain of the method is low, and opposite.

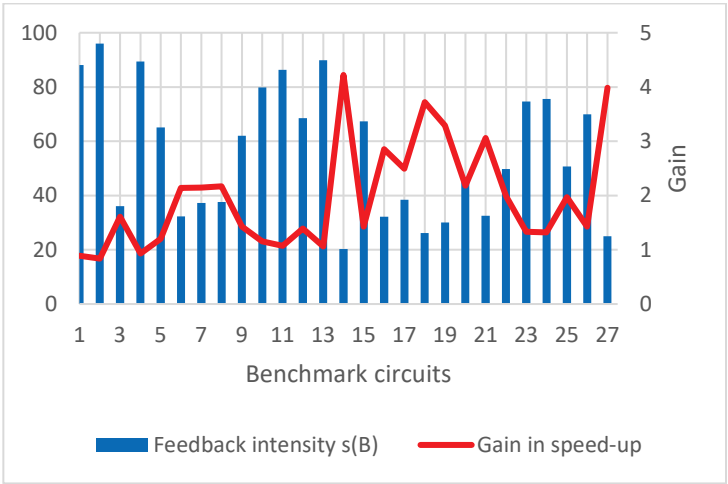


Figure 4.7: Dependence of gain on the feedback level

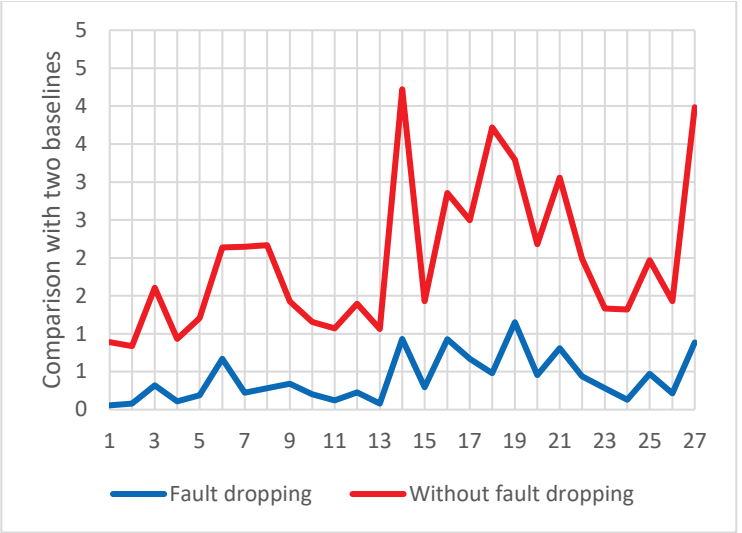


Figure 4.8: Comparison of the speed of the proposed method with two baselines: sequential fault simulation with and without fault dropping

Note, however, that the fault dropping based fault simulation approach is able to evaluate only the fault coverage of the given test, whereas the simulation without fault dropping provides not only the fault coverage data, but also the data for fault diagnosis purposes, e.g. the data needed for creating fault tables and fault dictionaries.

On the other hand, the main core of the proposed method – parallel critical path tracing of faults – is directed substantially to get diagnostic information for each test pattern separately; hence, it would be unfear to compare the proposed method with other simulation methods which use fault dropping approach.

Table 4.3: Dependence of the speed-up gain on the observability of flip-flops of the benchmark circuit b17

Design No	*s*(A), %	*s*(B), %	Observable FFs, #	Time costs, s			Gain
				t_{CP}	t_{CL}	t_{NEW}	
1	0,3	25,0	0	15,80	2,01	3227	4
2	3,8	21,5	200	15,71	2,03	631	20
3	6,6	18,7	400	15,61	1,94	551	23
4	8,4	16,9	600	15,73	1,88	500	26
5	9,8	15,5	800	15,93	1,88	460	28
6	15,0	10,3	1000	15,70	2,04	312	41
7	23,6	1,7	1200	15,69	2	65	197
8	25,3	0,0	1415	15,74	2,03	18	724

Figure 4.7 demonstrates that the high values of $s(B)$, which characterize the sequential level of the given circuit, work against the efficiency of the proposed method. The values of $s(B)$ will be high, when the ratio of the number of flip-flops and the number of primary outputs (#FF/#out) is very high. On the other hand, as can also be observed

from Figure 4.7, if the value of $s(B)$ is reduced then the speed of the proposed method will increase. This fact can motivate the redesign of sequential circuits for better observability to reduce the cost of testing, both, the test length and the time of fault simulation

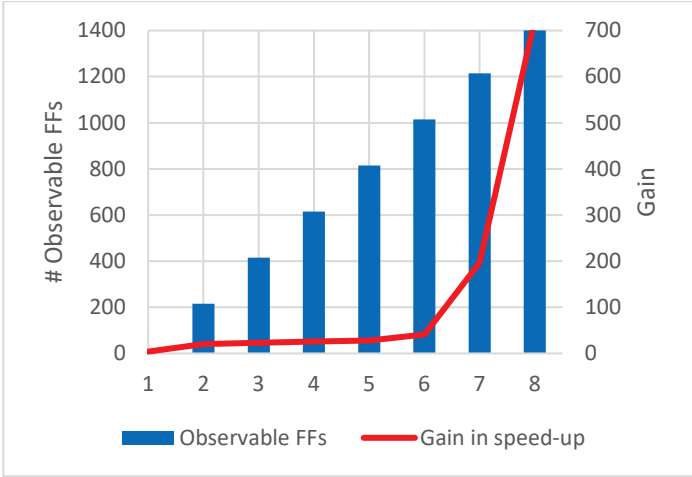


Figure 4.9: Relationship between the speed-up gain and the observability of flip-flops of the benchmark circuit b17

In Table 4.3, an experiment is presented with 8 different designs of a circuit b17 with different number of flip-flops made observable with additional primary outputs. The row 1 corresponds to the original circuit b17 used in Tables 4.1 and 4.2. A dramatic speed-up of simulation can be seen; the more flip-flops can be directly observable. The case in the row 8 corresponds to the circuit where all the flip-flops are observable.

As already mentioned, the results of the experimental research suggest a very fast procedure for prediction of the speed-up to be achieved by the new method compared to the conventional methods. Since $t_{anal} = t_{CP} + t_{CL} \ll t_{OLD}$, the best way of such prediction would be to carry out Algorithm 4.1 for calculating the set $S(B)$. The percentage of $S(B)$ from the full set of faults will be a good criterion (higher bound according to the formula (4-10)) about the expected efficiency of using the proposed method.

4.5 Conclusions

In this chapter a novel approach for fault simulation in sequential circuits is proposed, which allows achieving considerable speed-up in simulation time, compared to the conventional fault simulation methods used for sequential circuits. By experimental research, it was shown that for the pool of 27 benchmark circuits the average gain of speed-up was 2 times (in the best case even 5 times).

The high speed-up is achieved by integrating into the fault simulation process the exact parallel critical path tracing concept used so far only for fault simulation in combinational circuits.

The main novelties of the paper are as follows:

- A novel method was developed for separating the parallel critical path tracing into two parts: tracing to the primary outputs and to the pseudo-outputs (to feedback loops);
- A novel method was developed for classification of the faults into two classes: the faults directly detectable by combinational critical path tracing, and the faults to be simulated sequentially by traditional methods;
- The exact fault classification allowed integrating combinational and sequential methods into a joint fault simulation procedure for sequential circuits.

Also a very fast method was developed for pre-evaluating the applicability of the new fault simulation method, i.e. to predict the expected gain compared to the conventional methods. A dramatic speed-up can be expected if the value of $s(B)$ is small. In case of the investigated benchmark circuits the lowest value of $s(B)$ was 20% of faults from the full set of faults, which provided speed-up gain 5 times. Experiments showed that the decrease of $s(B)$ will improve the gain of the new method.

5 Simulating diagnostic experiments in digital circuits

Chapter 5 introduces a new application field for using SSBDDs for fault simulation purposes. A novel idea is proposed for representing Boolean differential equations in form of SSBDDs for simulating fault diagnosis processes in the general case of the multiple fault assumption (in the traditional practice, single fault assumptions are made). A method is proposed for converting Boolean differential equations as a model of diagnostic test experiment into SSBDDs where each node represents a signal transition as a faulty case. An algorithm was proposed to simulate the diagnostic experiments by manipulations of SSBDDs, where for manipulation of SSBDDs, a special 5-valued algebra was developed. A novel hierarchical approach for fault diagnosis of digital circuits is developed, where the role of the proposed SSBDD based diagnostic simulation method is to specify the fault location in ambiguous situations due to self-masking of multiple faults. Experimental results conclude the chapter.

The contribution of this chapter has been published in [96].

5.1 Modeling fault diagnosis with Boolean differential equations

Consider a single output combinational circuit with *n* inputs as a Boolean function $y = F(x_1, x_2, \dots, x_n)$. A test experiment with the circuit can be modeled as a Boolean full differential [42], [102]:

$$\begin{aligned} dy = y \oplus F\big((x_1 \oplus dx_1), (x_2 \oplus dx_2), \dots, (x_n \oplus dx_n)\big) = \\ = F(x_1, x_2, \dots, x_n) \oplus \\ \oplus F\big((x_1 \oplus dx_1), (x_2 \oplus dx_2), \dots, (x_n \oplus dx_n)\big) \end{aligned} \tag{5-1}$$

When applying a test pattern T_t, the diagnosis on the basis of this experiment can be represented as a logic assertion:

$$dy^t = F^t(dx_1^t, dx_2^t, \dots, dx_n^t) \tag{5-2}$$

where $dx_j^t \in \{dx_j, \overline{dx_j}\}$

$dx^t \in \{0,1\}$ denotes the test result: $dx^t = 0$, if the test pattern T_t has passed, and $dx^t = 1$, if the test pattern T_t has failed, dx_j means that there is a suspected fault related to x_j, and $\overline{dx_j}$ means that no fault at x_j is suspected. The fault type under question is defined by the value of x_j at the given pattern. To create the possibility of manipulations with faults of different types, let us introduce for each variable x_j a set of suspected diagnostic states:

$$DV(x_j) = \left\{dx_j^0, \overline{dx_j^0}, dx_j^1, \overline{dx_j^1}, \overline{dx_j}\right\} \tag{5-3}$$

where the assertions $dx_j^0, \overline{dx_j^0}, dx_j^1, \overline{dx_j^1}, \overline{dx_j}$ which may be true and false have the following meanings:

- the fault $x_j \equiv 1$ (stuck-at-1) is suspected (the upper index at dx_j means the value of x_j at the given test pattern),
- the fault $x_j \equiv 1$ (stuck-at-1) is not suspected,
- the fault $x_j \equiv 0$ (stuck-at-0) is suspected,

- the fault $x_j \equiv 0$ (stuck-at-0) is not suspected,
- and no faults are suspected at the variable x_j, respectively.

Let us introduce on the basis of (5-1) the following diagnostic equation as always a true assertion:

$$D(T_t) = \overline{dx^t} \oplus (y^t \oplus F^t) = 1 \tag{5-4}$$

Assuming a test experiment $T = (T_1, T_2)$ has been carried out with two test patterns, the following test responses (dy^1, dy^2) have been received, respectively. Thus, the statement about fault diagnosis $D(T)$ based on the test T can be calculated obviously from the logic multiplication of two assertions, which is always true

$$D(T) = D(T_1) \wedge D(T_2) = 1 \tag{5-5}$$

5.2 5-valued algebra for reasoning diagnostic processes

For processing the diagnosis equations of type (5-5), the following 5-valued algebra was developed as depicted in Table 5.1. The algebra allows to find out the inconsistencies of two assertions and to carry out the possible simplifications in the expressions of type (5-5).

Table 5.1: 5-valued algebra for calculating Boolean differentials

5-valued algebra					
dx	dx^0	$\overline{dx^0}$	dx^1	$\overline{dx^1}$	$\overline{dx}$
dx^0	dx^0	$\emptyset$	$\emptyset$	dx^0	$\emptyset$
$\overline{dx^0}$	$\emptyset$	$\overline{dx^0}$	dx^1	$\overline{dx}$	$\overline{dx}$
dx^1	$\emptyset$	dx^1	dx^1	$\emptyset$	$\emptyset$
$\overline{dx^1}$	dx^0	$\overline{dx}$	$\emptyset$	$\overline{dx^1}$	$\overline{dx}$
$\overline{dx}$	$\emptyset$	$\overline{dx}$	$\emptyset$	$\overline{dx}$	$\overline{dx}$

Denote by a_i – the entry in the leftmost column and in the i^{th} row, and by b_j – the entry in the top row and in the j^{th} row. Then, the entries in the rest of Table 5.1 mean the following logic assertions

$$e_{ij} = a_i \wedge b_j \tag{5-6}$$

The entries in Table 5.1 mean the conclusion to be drawn from two statements about the diagnostic status of a line in the circuit in terms of the meaning of the set of symbols (5-3).

The justifications of the entries e_{ij} in Table 5.1 are given as follows:

1. If a_i and b_j denote the same symbol, then $e_{ij} = a_i \wedge b_j$.
2. If a_i and b_j denote dx^0 and dx^1 (or, $dx^{1\prime}$ and dx^0), respectively, then the two statements are inconsistent, because the same line cannot be simultaneously (stuck-at-1) and (stuck-at-0). For the inconsistency of the two statements a_i and b_j let us use the notation $e_{ij} = \emptyset$.

3. if a_i and b_j denote dx^0 and $\overline{dx^0}$ (or, dx^1 and $\overline{dx^1}$), respectively, then the two statements are inconsistent, because the same line cannot be simultaneously faulty and not faulty. In this case, $e_{ij} = \emptyset$.

4. If a_i and b_j denote dx^0 and $\overline{dx^1}$ (or, dx^1 and $\overline{dx^0}$), respectively, then the statement is stronger than b_j , and $e_{ij} = a_i$. Similarly, if a_i and b_j denote $\overline{dx^1}$ and dx^0 (or, $\overline{dx^0}$ and dx^1), respectively, then the statement b_j is stronger than a_i , and $e_{ij} = b_j$.

5. If a_i and b_j denote $\overline{dx^0}$ and $\overline{dx^1}$ (or, $\overline{dx^1}$ and $\overline{dx^0}$), respectively, then the statements a_i and b_j complement each other regarding the absence of both faults, stuck-at-1 and stuck-at-0, which means that the line is not faulty. For this case, the notation $e_{ij} = \overline{dx}$ is used

Using the above introduced algebra for reasoning the statements about the diagnostic state of the lines in the given circuit, the diagnostic equations of type (5-5) can be simplified step by step for all the test experiments with all test patterns, and then, come up with the final statement of the fault diagnosis. If the final reduced assertion $D(T)$ will consist of a single DNF term, the diagnosis statement is unambiguous. More than one term will mean ambiguity. The more test patterns are used in the test experiment, the less ambiguous the diagnosis becomes.

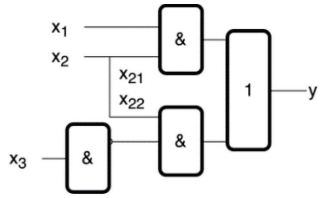


Figure 5.1: Combinational circuit

Algorithm 5.1 of fault diagnosis

The fault diagnosis using the algebra in Table 5.1 is carried in the following way (the procedure is illustrated in Example 5.1 and in Table 5.2).

1. A set of test patterns $T_t = \{T_t\}$ is carried out, and for each test pattern $T_t \in T$, the response $dy^t \in \{0,1\}$ is fixed.

2. For each $T_t \in T$, the diagnostic equation of type (5-4) is fixed.

3. For the first two test patterns, the diagnostic equation of type (5-5) is fixed and simplified using the algebra in Table 5.1. The simplified equation (true statement) represents the current fault diagnosis to be updated and improved (made more accurate with less ambiguity), using the test experiment results with other test patterns.

4. A new diagnostic equation of type (5-5) is created, using the diagnostic equation of the current diagnosis and the diagnostic equation for the next test pattern. The equation is simplified using the algebra of Table 5.1, and as the result, a new current fault diagnosis is created.

5. The step (4) of the algorithm is repeated step-by-step for all test patterns involved in the test experiment. The resulting equation will represent the current diagnosis for the given circuit.

Example 5.1

Consider a circuit in Figure 5.1, and the reasoning of the test experiment according to Algorithm 5.1, illustrated in Table 5.2. The left part of the table gives information about the test patterns (the values of input variables, and the expected values of output responses for the circuit). The table illustrates the diagnostic process for the case where all 5 test patterns pass. In this case, each passed test experiment for the related test pattern will produce the response $dy = 0$. In the rightmost column of Table 5.2 the diagnostic equations of type (5-4) for all five test patterns are depicted.

Table 5.2: Diagnostic process with 5 passed test patterns

T_t	x_1	x_2	x_3	y	$D(T_i)$	Diagnostic assertions
T_1	0	1	1	0	$D(T_1)$	$(\overline{dx_1^0} \vee dx_{21}^1)(dx_{22}^1 \vee \overline{dx_3^1}) = 1$
T_2	1	1	1	1	$D(T_2)$	$\overline{dx_1^1 dx_{21}^1} \vee \overline{dx_{22}^1} dx_3^1 = 1$
$D_2=D(T_1,T_2)$						$\overline{dx_1}\,\overline{dx_{21}^1}(dx_{22}^1 \vee \overline{dx_3^1}) = 1$
T_3	1	0	1	0	$D(T_3)$	$(dx_1^1 \vee \overline{dx_{21}^0})(\overline{dx}_{22}^{0} \vee \overline{dx_3^1}) = 1$
$D_3=D(T_1,T_2,T_3)$						$\overline{dx_1 dx_{21}}(\overline{dx_3^1} \vee \overline{dx_{22}^0} dx_{22}^1) = 1$
T_4	0	1	0	1	$D(T_4)$	$dx_1^0 \overline{dx_{21}^1} \vee \overline{dx_{22}^1 dx_3^0} = 1$
$D_4=D(T_1,T_2,T_3,T_4)$						$\overline{dx_1 dx_{21} dx_3}\,\overline{dx_{22}^1} = 1$
T_5	0	0	0	0	$D(T_5)$	$(\overline{dx_1^0} \vee \overline{dx_{21}^0})(\overline{dx_{22}^0} \vee dx_3^0) = 1$
$D_5=D(T_1,T_2,T_3,T_4,T_5)$						$\overline{dx_1 dx_{21} dx_{22} dx_3} = 1$

After the first two test experiments with two test patterns, the diagnostic equation $D_2 = D(T_1, T_2)$ of type (5-5) is created and simplified using the algebra in Table 5.1. The obtained results $\overline{dx_1}$=1 and $\overline{dx_{21}^1}$=1 state unambiguously that the signal path from the input x_1 up to the output y is working correctly, and the fault $x_{21} \equiv 0$ is missing. After the third test, it becomes clear that the fault $x_{21} \equiv 1$ is missing as well. After the 5th test, it can be stated that the circuit is functioning correctly – there are no faults on any paths of the circuit.

5.3 Simulating diagnostic experiments with diagnostic SSBDDs

BDDs have been proved to be an efficient tool for manipulations with Boolean functions [5]. Since the diagnostic equations (5-4) represent Boolean expressions and the diagnostic processing of test results according to (5-5), the final diagnostic assertions by manipulations with BDDs can be easily found, which allows to avoid the explosion of the expressions of type (5-4) and (5-5).

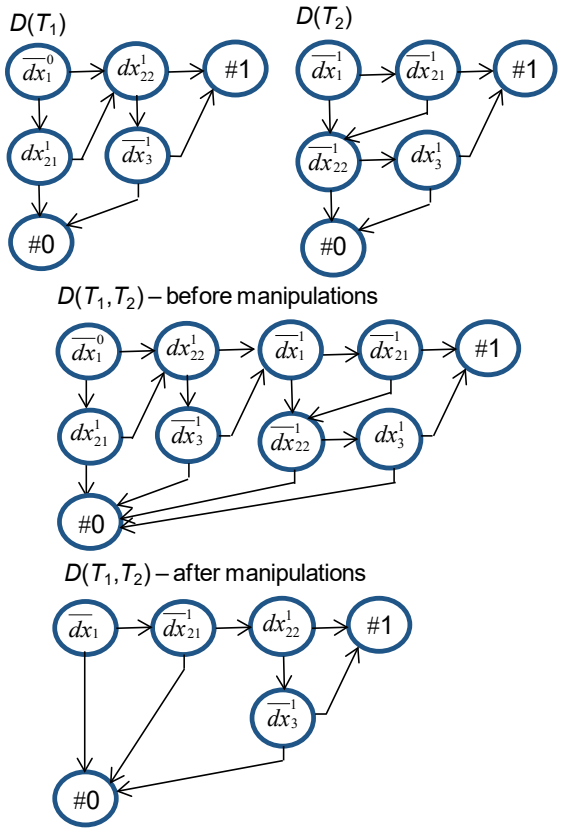


Figure 5.2: SSBDDs for diagnostic experiment $D(T_1, T_2)$)

Since the fault diagnosis has to be made in a close connection to the structure of the circuit under diagnosis, the Structurally Synthesized BDDs have to be used, as discussed in Section 1.1. As an example, in Fig. 6-2, the SSBDDs for the three diagnostic equations $D(T_1)$, $D(T_2)$, and $D(T_1, T_2)$, presented in Table 6-1, are shown. Finally, the simplified SSBDD for $D(T_1, T_2)$ is as well depicted.

Algorithm 5.2 of fault diagnosis with SSBDDs
Consider a diagnostic experiment based on the test set $T = \{T_1, T_2, \ldots, T_n\}$.

1. For modeling the faults in the expressions $D(T_k)$, $T_k \in T$, Structurally Synthesized BDDs (SSBDD) are created. The BDD for $D(T_1)$ will represent the first current diagnosis D_1.
2. Iteratively, each next diagnosis D_k after the experiment with test T_k will be calculated as $D_k = D_{k-1} \wedge D(T_k)$.
3. First, the joint SSBDD will be created for D_k, where each 1-path in the SSBDD corresponds to a term in the DNF of D_k represented by SSBDD.
4. Next, all the 1-paths in the SSBDD will be processed, and the inconsistent paths, in accordance to the 5-valued algebra in Table 5.1, are excluded from the SSBDD.

5. The steps 2-4 are carried out step-by-step for all test patterns in T. The final simplified SSBDD for $D(T_k)$ represents the statement of fault diagnosis derived from the given test experiment.

Example 5.2

Consider in Fig. 5.2 the two SSBDDs for representing $D(T_1)$ and $D(T_2)$ in Table 5.2 (in the first two rows). The labels on the edges of the SSBDDs are omitted, the right-hand edge from a node corresponds to the value 1, and the down-hand edge corresponds to the value 0 of the node variable. The graph for $D(T_1)$ contains four 1-paths, the graph for $D(T_2)$ two. The merged SSBDD for $D(T_1, T_2)$ contains eight 1-paths. By processing the paths to simplify the SSBDD according to the algebra in Table 5.1, 6 inconsistent ones are excluded, and SSBDD with only two 1-paths are created, which represents the statement of two possible diagnostic cases $\overline{dx_1}\overline{dx_{21}^1} = 1$ or $dx_{22}^1\overline{dx_3^1} = 1$.

The developed new method of solving Boolean differential equations using SSBDDs can be regarded as a new field of application of SSBDDs where the nodes are representing the Boolean differentials of Boolean variables instead of simply Boolean variables as it has been the tradition in the past.

Let us call the SSBDDs with nodes labeled by Boolean differentials of Boolean variables as Diagnostic SSBDDs (DSSBDD). Each such a DSSBDD can be used for solving fault diagnosis problems in corresponding digital circuits under assumption of the presence of any multiple fault combination. The self-masking possibilities of multiple faults are taken into account in the model automatically.

The novel algebra developed in this Chapter for simplification of Boolean differential equations can be directly used also for manipulation and simplification of SSBDDs.

5.4 Hierarchical fault diagnosis with test groups

In [80]-[81] a conception of test groups was introduced, and the necessary conditions for detecting MSAF in combinational circuits were introduced. The goal of a test group is to verify the correctness of a selected part of the circuit. In case of passing of all the test groups, the circuit is proven fault-free. In case when not all test groups will pass, a fault diagnosis is needed, which can be carried out by solving diagnostic equations locally, i.e. in the selected region of the circuit targeted by the test group?

For this local fault diagnosis, the proposed method of simulating diagnostic experiments with Boolean differential equations and corresponding SSBDDs can be used.

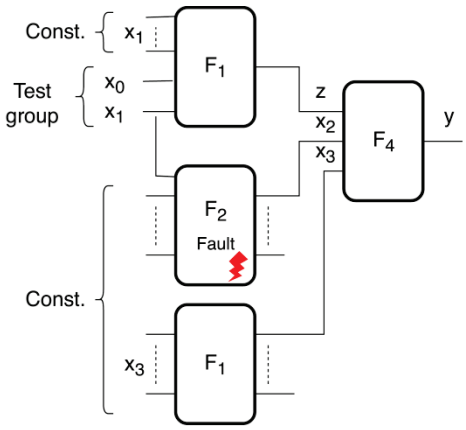


Figure 5.3: Hierarchical fault diagnosis

Consider a circuit in Fig. 5.3 as a higher level network with blocks F_1, F_2, F_3, and F_4. Let us start the fault diagnosis first, at the lower level, by selecting a sub circuit with inputs x_0 and x_1 as a target in the block F_1. The sub circuit will be tested along an activated test path L through the wire z up to the output y of the circuit.

Assuming there is a test group $TG = \{T_0, T_1, T_3\}$ applying to the inputs x_0 and x_1 a set of changing patterns whereas the values on all other primary inputs of the circuit are kept constant, according to the test group conception [80]-[81]. The role of the test group is to concentrate on testing with all the patterns T_0, T_1, T_3 only on proving the correctness of the joint signal path L. If the test group will pass, it can be concluded that the path L is working correctly. Each additional passed test group will add information about other correct signal paths. This information will help to locate the faults if some test groups will fail.

If the correctness of a subset S of signal paths has been proven, then these paths can be used for sending correct signals to the needed connections on the higher network level which makes it easier to test other blocks at the lower level. For example, if correct signals can be forced on the wires z and x_3 in Figure 5.3, to activate a test path from x_2 up to the output y, the fault reasoning might only be carried out in the block F_2 locally, to reduce the complexity of the diagnosis problem.

Example 5.3

Consider the circuit in Figure 5.3, and apply the test group $TG = \{T_0, T_1, T_3\}$ to the block F_1, so that the inputs x_0 and x_1 will have the local patterns $\{11,01,10\}$ and the values on all other primary inputs of the circuit are kept constant. Assume that the activated by TG test path can be represented by a function:

$$y = F(x_0, x_1, x_2, x_3, X_1) \tag{5-7}$$

Assume also that all the signals activating the test path of the test group and originating at the inputs X_1 have been proved as correct. This allows reducing the diagnosis problem raised by the test group TG to processing of the simplified function:

$$y = x_0 x_1 \vee x_2 \vee x_3 \tag{5-8}$$

The diagnostic process with three passed test patterns of the test group is depicted in Table 5.3. The final assertion , $D(T_0, T_1, T_2)$ states that no faults are present along the signal paths from the inputs x_0 and x_1 up to the output *y*, and the faults $x_2 \equiv 1$ and $x_3 \equiv 1$ are missing on the inputs of the block F_4. The knowledge about the correctness of the wire *z* and the missing fault $x_3 \equiv 1$ allows carrying out the fault reasoning in the block F_2 only locally.

Table 5.3: Diagnostic processes for the circuit in Fig.3

T_t	x_0	x_1	x_2	x_3	y	dy	Assertions
T_0	1	1	0	0	1	0	$\overline{dx_0}^1 \overline{dx_1}^1 \vee dx_2^0 \vee dx_3^0$
T_1	0	1	0	0	0	0	$(\overline{dx_0}^0 \vee dx_1^1)\overline{dx_2}^0 \overline{dx_3}^0$
$D(T_0, T_1)$							$\overline{dx_0}\, \overline{dx_1}^1 \overline{dx_2}^0 \overline{dx_3}^0$
T_2	1	0	0	0	0	0	$(dx_0^1 \vee \overline{dx_1}^0)\overline{dx_2}^0 \overline{dx_3}^0$
$D(T_0, T_1, T_2)$							$\overline{dx_0}\, \overline{dx_1}\, \overline{dx_2}^0 \overline{dx_3}^0$
T_2'	1	0	0	0	0	1	$\overline{dx_0}^1 dx_1^0 \vee dx_2^{0*} \vee dx_3^0$
$D(T_0, T_1, T_2$')							$\overline{dx_0}\, \overline{dx_1}^1 \overline{dx_3}^0 (dx_1^0 \vee dx_2^{0*} \overline{dx_2}^0)$

Suppose the pattern $T_2' \in TG$ will fail. The diagnostic statement $D(T_0, T_1, T_2')$ refers to either the fault candidate $x_2 \equiv 1$ or to the unstable behavior of the wire x_2 during execution of the test group. The reason of the instability of x_2 may be a fault in the block F_2 which because of the changing value of x_1 may sometimes influence on x_2, and sometimes not. In this case the test group has not fulfilled its role to prove the correctness of the wire z.

In this case, the proposed new method has an important contribution. The role of the Boolean full differential as shown in Example 3.1, lays on specifying the fault candidates in the case when the test group will fail. The method of solving differential equations will help also in this case when some of the test groups cannot be synthesized and it would not be possible to prove the correctness of some parts of the circuit.

5.5 Experimental results

In Table 5.4, experimental data are presented regarding the test group synthesis for the ISCAS'85 benchmark circuits [100]. The columns 3 and 4 show the number of patterns in the single stuck-at-fault (SSAF) test and in the multiple stuck-at-fault (MSAF) tests, respectively. The 3-pattern test groups were built for the gates of the circuits whereas many test groups were possible to merge. Repeated patterns were removed from the test set. Still, the synthesis of test groups resulted in the test sets many times larger compared to the traditional SSAF tests. The fair comparison between the SSAF and MSAF test lengths, however, cannot be done in the present research, since the test groups for different outputs were not merged. Merging of test groups will provide a significant reduction of the MSAF test length.

Table 5.4: Experimental data of generating test groups for multiple fault diagnosis

Circ.	Gates #	SSAF test #	MSAF test #	Group cover %
c432	275	53	314	82,91
c499	683	86	482	67,2
c880	429	84	546	99,8
c1355	579	86	514	65,6
c1908	776	123	621	96,3
c2670	1192	103	820	76,3
c3540	1514	148	995	80,3
c5315	2240	104	1523	91,8
c6288	2480	22	465	98,1
c7552	3163	202	1863	87,8

The test group coverage means the percentage of the test groups that were built successfully. The test group coverage characterizes the feasibility of the concept to prove the correctness of subcircuits instead of targeting the faults to be tested.

For diagnosing the subcircuits not covered by test groups, the new proposed method of calculation the Boolean full differentials using SSBDDs can be used.

5.6 Conclusions

In this chapter the fault simulation was investigated as a possible answer to solve two sides of the fault diagnosis problem: how to develop efficient diagnostic test sequences and how to locate the faults if some test patterns will not pass. The recently proposed test group concept [81], [103] allows coping with fault masking, and it was combined in this Thesis to improve the accuracy of fault diagnosis.

The problem of fault diagnosis in a general case, where any combination of multiple faults may be present, has no approved practical solution so far. To contribute to this unsolved problem, in this Thesis it was proposed to simulate the test experiments with Boolean differential equations, and to solve these equations using SSBDDs supported by the novel 5-valued algebra.

The role of test groups is to prove step by step the correctness of the given circuit. If the test groups cover the whole circuit, and all the test groups will pass, the fault reasoning is not needed any more, and the circuit can be stated as correct and free of faults. If some parts of the circuit cannot be exercised by test groups or faults have been detected, the fault reasoning by solving Boolean differential equations is needed to cope with the multiple SAF (MSAF) case. As the experiments showed, it was not possible to synthesize in average 15% of 3-pattern test groups for the gates. The proposed new method of using Boolean full differentials should be used if the test groups cannot be created, and it can be used also for reasoning MSAF when some of the test groups will not pass during test.

The developed new method of solving Boolean differential equations using SSBDDs can be regarded as a new field of application of SSBDDs where the nodes are representing the Boolean differentials instead of Boolean variables as it has been the tradition in the past. The novel algebra developed in this Chapter for simplification of Boolean differential equations can be directly applied also for manipulation and simplification of SSBDDs.

6 Conclusions, results and future work

6.1 Conclusions

The main focus of the research work on the Thesis has been on developing new methods for fault simulation in digital circuits.

The target has been to increase the speed of algorithms and software tools for fault simulation as crucial components of the toolboxes of CAD for testing of digital circuits, and to contribute for shortening the time to market in the VLSI industry. Another target has been to improve as well the accuracy of evaluating the quality of test sequences in terms of fault coverage.

The focus of the Thesis and the contribution of the research results to state-of-the-art are illustrated in a compact way in Figure 6.1.

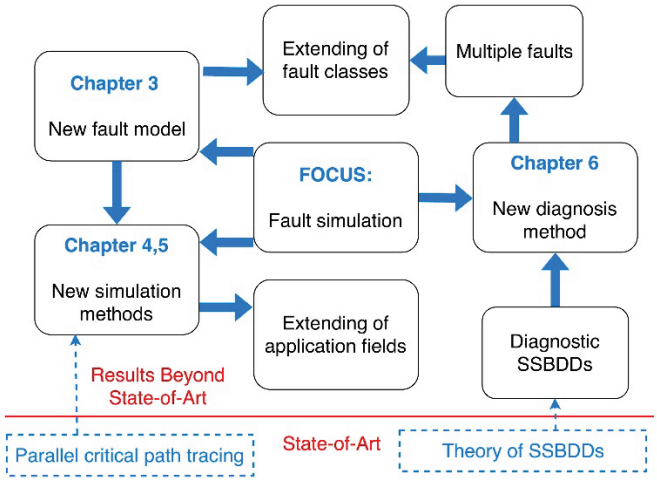


Figure 6.1: Focus of the Thesis and the contributions beyond State-of-the-Art

The main input for the research was the current state of the art in fault simulation, particularly:

- the concept of a very fast parallel exact critical path tracing for stuck-at-fault reasoning in combinational circuits, and
- the theory of Structurally Synthesized BDDs (SSBDD) as a concept of integrating both, structural and functional aspects of combinational circuits into a concise graph model.

The main outputs of the research carried out in the Thesis (contributions beyond state-of-the-art) are:

- extension of the fault class for timing simulation of digital circuits, particularly, by introducing novel types of transition delay fault models (the content of Chapter 2);
- extension of the concept of parallel critical path fault tracing for applying it instead of only for combinational circuits also for sequential circuits (the contents of Chapters 3 and 4);

- extension of the concept of SSBDDs by introducing a novel type of diagnostic SSBDDs for simulating diagnostic experiments of digital circuits (the content of Chapter 5).

6.2 Main scientific results

The main results of the research, formulated also in Section 3 in Introduction, can be specified in context of Figure 6.1 in the following way:

1. A new fault model (non-robust functionally sensitized TDF), and a new delay fault simulation method was introduced to improve the quality of delay fault testing, compared to state-of-the-art.

   The new model is an important extension of the traditional class of delay faults. To increase the speed of delay fault simulation, two novel 7-valued algebras were developed and combined with exact parallel critical path fault tracing, used so far only for the class of stuck-at-faults (contribution of Chapter 2).

2. The first time, a method was developed, which allows exact parallel fault tracing for fault simulation in sequential circuits with dramatic speed-up of fault analysis, compared to traditional methods of fault simulation.

   The method was implemented in two versions of cutting off the global feedback loops: the first, on the basis of using minor design for testability features (contribution of Chapter 3), and the second, on the basis of virtual cut-off of feedback loops (contribution of Chapter 4).

3. The first time, a method is proposed for fault diagnosis in digital circuits in general case of multiple faults (contribution of Chapter 5).

   For that purpose, a new type of Diagnostic SSBDDs was introduced for simulating diagnostic experiments represented as Boolean differential equations. It was also shown how the new method can be used in hierarchical approaches to fault diagnosis in complex circuits.

6.3 Future work

As future work the following research can be pursued:

- The new types of transition delay faults developed in the Thesis (Chapter 2) have been used in the new delay simulation method and algorithms developed and investigated here for combinational circuits. The possibilities of extending their use also for the fault simulation methods developed for sequential circuits (Chapter 3 and 4), can be investigated in the future research.
- The new type of Diagnostic SSBDDs developed in Chapter 5 was exploited in this Thesis for simulation of diagnostic experiments in case where the test sequences were already available. It could be worthy to further investigate the power of this model in the field of generating (synthesis) of test sequences.

List of Figures

List of Tables

References

[1] M.L. Bushnell, V.D. Agrawal (2013). Essentials of Electronic testing. Kluwer Acad. Publishers.

[2] H.-J. Wunderlich, Ed. (2010). Models in Hardware Testing. Springer.

[3] L.-T. Wang, C.-W. Wu, X. Wen (2006). VLSI Test Principles and Architectures. Elsevier.

[4] D. Gizopulos (2006). Advances in Electronic Testing, Technology & Engineering. Springer.

[5] Bryant, R.E. (1986). Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Computers (Vol.C-35), No 8, pp. 667-690

[6] Minato, S. (1996). BDDs and Applications for VLSI CAD. Kluwer Academic Publishers.

[7] Karpovsky, M.G., Stanković, R.S., Astola J.T. (2008). Spectral Logic and Its Applications for the Design of Digital Devices. Wiley-Interscience.

[8] Ubar, R. (1976) Test Generation for Digital Circuits with Alternative Graphs. Proceedings of Tallinn Technical University, No 409, pp. 75-81. (in Russian)

[9] Ubar, R. (1996). Test Synthesis with Alternative Graphs. IEEE Design&Test of Computers, Spring, pp. 48-57.

[10] R. Ubar, J. Raik, T. Vierhaus (2011). Design and Test Technology for Dependable Systems-on-Chip. IGI Global, USA.

[11] Krstić, Kwang-Ting (Tim) Cheng (1998). Delay Fault Testing for VLSI Circuits. Kluwer Academic Publishers. Boston/Dordrecht/London.

[12] R. Ubar, L. Jürimägi, E. Orasson, J. Raik (2016). Fault Collapsing in Digital Circuits Using Fast Fault Dominance and Equivalence Analysis with SSBDDs. In "VLSI-SoC: Design for Reliability, Security, and Low Power," Eds. Y. Shin, C. Tsui, J. Kim, K. Choi, R. Reis. Springer, pp. 23-45.

[13] Lee, C.Y. (1959). Representation of Switching Circuits by Binary Decision Programs. The Bell System Technical Journal, pp. 985-999.

[14] Sasao, T., Fujita, M. (eds.) (1996). Representations of Discrete Functions. Kluwer Academic Publishers.

[15] Drechsler, R., & Becker, B. (1998). Binary Decision Diagrams. Kluwer Academic Publishers.

[16] Minato, S, Ishiura, N., & Yajima. (1990). Shared binary decision diagrams with attributed edges for efficient Boolean function manipulation. Proc. 27th IEEE/ACM ICCAD'90, pp. 52-57.

[17] Srinivasan, A., Kam, T., Malik, Sh., & Bryant. (1990). Algorithms for discrete function manipulation. Proc. of Informations Conference on CAD, ICCAD-90, pp. 92-95.

[18] Kebschull, U., Schubert, E., & Rosenstiel, W. (1992). Multilevel logic synthesis based on functional decision diagrams. IEEE EDAC'92.

[19] Minato, S. (1995). Zero-suppressed BDDs for set manipulation in combinational problems. Proc. 30th ACM/IEEE DAC, pp. 272-277.

[20] Bahar, R.I., Frohm, E.A., Gaona, C.M., Hachtel, G.D., Macii, E., Pardo, A., & Somenzi, F. (1993). Algebraic decision diagrams and their applications. Int. Conf. on CAD, pp. 188-191.

[21] Bern, J., Meinel, C., & Slobodova, A. (1995). Efficient OBDD-based manipulation in CAD beyond current limits. 32-nd Conference on Design Automation, pp. 408-413.

[22] Clarke, E.M., Fujita, M., & Zhao, X. (1996). Multi-terminal binary decision diagrams and hybrid decision diagrams, In: Sasao, T., & Fujita, M. (eds.), Representations of Discrete Functions, Kluwer Academic Publishers, pp. 93-108.

[23] Ubar, R. (1998). Multi-Valued Simulation of Digital Circuits with Structurally Synthesized Binary Decision Diagrams. OPA, N.V. Gordon and Breach Publishers, Multiple Valued Logic (Vol. 4), pp. 141-157.

[24] J. Raik, R. Ubar, S. Devadze, A. Jutman. Efficient Single-Pattern Fault Simulation on Structurally Synthesized BDDs. Lecture Notes in Comp. Science, Vol. 3463, Springer Verlag, Berlin, Heidelberg, New York 2005, pp. 332-344.

[25] R. Ubar, S. Devadze, J. Raik, A. Jutman. Fast Fault Simulation in Digital Circuits with Scan Path. 13th Asia and South Pacific Design Automation Conference – ASP-DAC 2008, Seoul, Korea, Jan. 21-24, 2008, pp. 667-672.

[26] R. Ubar, S. Devadze, J. Raik, A. Jutman. Fast Fault Simulation for Extended Class of Faults in Scan-Path Circuits. 5th IEEE Int. Symposium on Electronic Design, Test and Applications – DELTA 2010. Ho Chi Minh City, Vietnam, Jan. 13-15, pp. 14-19.

[27] 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.

[28] R. Ubar. Fault Diagnosis in Digital Devices. Proceedings of the Estonian Academy of Sciences, Engng., 1995, No. 1/1, pp. 51-67.

[29] R. Ubar. Design Error Diagnosis with Resynthesis in Combinational Circuits. Journal of Electronic Testing: Theory and Applications 19, 73-82, 2003. Kluwer Academic Publishers.

[30] R. Ubar, J. Heinlaid, J. Raik, L. Raun. Calculation of Testability Measures on Structurally Synthesized Binary Decision Diagrams. Proc. of the 6th Baltic Electronics Conference, Oct. 7-9, 1998, Tallinn, pp. 179-182

[31] R. Ubar, T. Vassiljeva, J. Raik, A. Jutman, M. Tombak, A. Peder. Optimization of Structurally Synthesized BDDs. The 4th IASTED International Conference on Modelling, Simulation and Optimization, Kauai, Hawaii, USA, August 17-19, 2004, pp. 234-240.

[32] J.A. Waicukauski, et.al. Fault Simulation of Structured VLSI. *VLSI Systems Design*, Vol. 6, No. 12, pp. 20-32, 1985.

[33] B. Underwood, J. Ferguson. The Parallel Test Detect Fault Simulation Algorithm. *ITC*, pp. 712-717, 1989.

[34] 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.

[35] 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.

[36] 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.

[37] D. Harel, R. Sheng, J. Udell. Efficient Single Fault Propagation in Combinational Circuits. *Int. Conf. on CAD*, pp. 2-5, 1987.

[38] D.B. Armstrong. A deductive method for simulating faults in logic circuits. *IEEE Trans. Comp*., C-21(5), 464-471, 1972.
[39] E.G. Ulrich, T. Baker. Concurrent simulator of nearly identical digital networks. IEEE Trans.on Comp.,7(4), pp. 39-44, 1974.
[40] W.T. Cheng, M.L. Yu. Differential fault simulation: a fast method using minimal memory. *DAC*, pp. 424-428, 1989.
[41] 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.
[42] R. Ubar, S. Devadze, J. Raik, A. Jutman. Ultra Fast Parallel Fault Analysis on Structural BDDs. ETS, Freiburg, May 20-24, 2007.
[43] A. Thayse, M. Davio. "Boolean Differential Calculus and its Applications to Switching Theory". IEEE Trans. Comput. V.C-22, No. 4, pp. 409-420, 1973.
[44] F.J.O. Diaz. Fault Masking in Combinatorial Logic Circuits. IEEE Trans. on Computers, vol. C-24, no. 6, pp. 476-482, 1975.
[45] Thayse, "Boolean Calculus of Differences", Springer Verlag, 1981.
[46] J.A. Waicukauski, E. Lindbloom, B.K. Rosen, V.S. Iyengar, "Transition fault simulation", *IEEE Design and Test*, pp. 32-38, April 1987.
[47] G.L. Smith, "Model for delay faults based upon paths", in *Proc. Int. Test Conf.*, 1985, pp. 342-349.
[48] R.L. Wadsack et. al, "CMOS IC stuck-open fault electrical effects and design considerations", in *Proc. of IEEE ITC*, 1989, pp. 423-430.
[49] I. Pomerantz, M. Reddy, "Transition path delay faults: a new path delay fault model for small and large delay defects", *IEEE Trans. On VLSI Systems,* Vol. 16, No. 1, 2008, pp. 98-107.
[50] K. Heragu, J.H. Patel, V.D. Agrawal, "Segment delay faults: A new fault model", in *Proc. 14th VLSI Test Symp*., 1996, pp. 32-39.
[51] M. Sharma, J.H. Patel, "Testing of critical paths for delay faults", in *Proc. Int. Test. Conf*., 2001, pp. 634-641.
[52] J.V. Deodhar, S. Tragoudas, "Implicit deductive fault simulation for complex delay fault models", *IEEE Trans. on VLSI Systems*, Vol. 12, No. 6, 2004, pp. 636-641.
[53] I. Pomerantz, "Generation of mixed test sets for transition faults", IEEE Trans. On VLSI Systems, Vol. 20, No. 10, 2012, pp. 1895-1899.
[54] I. Pomerantz, M. Reddy, J.H. Patel, "On double transition faults as a delay fault model", in *Proc. Great Lakes Symp. VLSI*, 1996, pp.282-287.
[55] P. Gupta, M.S. Hsiao, "ALAPTF: A new transition fault model and the ATPG algorithm", in *Proc. of IEEE Int. Test Conf.,* 2004, pp.1053-1060.
[56] N. Jha, S. Gupta, *Testing of Digital Systems*. Cambridge, UK, Cambridge Univ. Press, 2003.
[57] I. Pomeranz, S.M. Reddy, "Hazard-Based Detection Conditions for Improved Transition Path Delay Fault Coverage", *IEEE Trans. On CAD of Integrated Circuits and Systems*, Vol. 29, No. 9, 2010, pp. 1449-1453.
[58] I. Pomeranz and S.M. Reddy, "Hazard-based detection conditions for improved transition fault coverage of scan-based tests", IEEE Trans. Very Large Scale Syst., Vol. 18, No. 2, 2010, pp. 333–337.
[59] F. Brglez, D. Bryan, K. Kominski, "Combinational Profiles of Sequential Benchmark Circuits," Int. Symp. on Circuits and Systems, 1989, pp. 1929-1934.

[60] F. Corno, M.S. Reorda, G. Squillero, “RT-level ITC'99 Bench-marks and First ATPG Results,” In Proc. Of the IEEE Design & Test of Computers, Vol. 17, No. 3, 2000, pp. 44-53.
[61] V. Agarwal, A. Fung. Multiple Fault Testing of Large Circuits by Single Fault Test Sets. IEEE Trans. on Comp, C-30, no. 11, pp. 855-865, 1981.
[62] K. Kubiak, W.K. Fuchs. Multiple-Fault Simulation and Coverage of Deterministic Single-Fault Test Sets. Int. Test. Conf., 1991, pp. 956-962.
[63] Y. Karkouri, E.M. Aboulhamid, E. Cerny. Test pattern Generation for Multiple Stuck-at-Faults. ETC’93, 1993, pp. 230-239.
[64] J.L.A. Hughes. Multiple Fault Detection Using Single Fault Test Sets. IEEE Trans. on CAD, vol. 7, no. 1, 1988, pp. 100-108.
[65] M. Abramovici, M.A. Breuer. Multiple Fault Diagnosis in CCs Based on an Effect-Cause Analysis. IEEE Trans.C-.29, 1980, pp. 451-460.
[66] Y. Karkouri, E.M. Aboulhamid, E. Cerny, A. Verreault. Use of Fault Dropping for Multiple Fault Analysis. IEEE Trans. on Computers, vol. 43, 1994, pp. 98-103.
[67] H. Cox, J. Rajski. A Method of Fault Analysis for Test Generation and Fault Diagnosis. IEEE Trans. on CAD, vol. 7, no. 7, 1988, pp. 813-833.
[68] J. Jacob, V.D. Agrawal. Multiple Fault Detection in 2-Level Multioutput Circuits. J. Electronic Testing, Vol. 3, No. 2, 1992, pp. 171-173.
[69] P. Camurati, et al. “Improved Techniques for Multiple Stuck-at Fault Analysis Using Single Stuck-at Fault Test Sets”, Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS ’92), May 1992, Vol. 1, pp. 383-386.
[70] M. Hunger, S. Hellebrand. Verification and Analysis of Self-Checking Properties through ATPG. On-Line Test Symp. 2008, pp. 25-30.
[71] M. Hunger, S. Hellebrand. ATPG-Based Grading of Strong Fault-Secureness. 15th IEEE On-Line Testing Symposium, 2009, pp. 269-274.
[72] Y.C. Kim, V.D. Agrawal, K.K. Saluja. Multiple Faults: Modeling, Simulation and Test. 15th Int. Conf. on VLSI Design, 2002, pp. 1-6.
[73] Y. Zhao, Y. Li. A Multiple Faults Test Generation Algorithm Based on Neural Networks and Chaotic Searching for Digital Circuits. Conf. Computational Intelligence and Software Eng. – CiSE, 2010, pp. 1-3.
[74] E. Macii, T. Wolf. Multiple Stuck-at Fault Test Generation Techniques for Combinational Circuits Based on Network Decomposition. 36th Midwest Symp. on CAS, 1993, vol. 1, pp. 465-467.
[75] I. Pomeranz, S.M. Reddy. On Generating Test Sets that Remain Valid in the Presence of Undetected Faults. 7th Great Lakes Symp. on VLSI. 1997, pp. 20-25.
[76] H. Takahashi, N. Iuchi, Y. Takamatsu. Test generation for Combinational Circuits with Multiple Faults. Pacific Rim International Symp. on Fault-Tolerant Systems. 1991, pp. 212-217.
[77] S. Kajihara, R. Nishigaya, T. Sumioka, K. Kinoshita. Efficient Techniques for Multiple Fault Test Generation. 3rd ATS, 1994, pp. 52-56.
[78] A. Agrawal, A. Saldanha, L. Lavagno. Compact and Complete Test Set Generation for Multiple Stuck-at Faults. ICCAD’93, 1996, pp. 212-219.
[79] A.G. Birger, E.T. Gurvitch, S. Kuznetsov. Testing of Multiple Faults in Comb. Circuits. Avtomatika i Telemehanika, No 8, 1975, pp. 113-120.
[80] R. Ubar, S. Kostin, J. Raik. “About Robustness of Test Patterns Regarding Multiple Faults”. 13th IEEE Latin American Test Workshop, Quito, Ecuador, April 10-13, 2012, pp. 86-91.

[81] R. Ubar, S. Kostin, J. Raik. “Multiple Stuck-at-Fault Detection Theorem”. The 15th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, Tallinn, Estonia, April 18-20, 2012, pp. 236-241.

[82] J. Kõusaar, R. Ubar. About testing of transition delay faults. Proceedings of the Workshop BELAS, Cottbus, Germany, 2014.

[83] 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, 2014.

[84] J. Kõusaar, R. Ubar. 7-Valued Algebra for Transition Delay Fault Analysis. Proceedings of Baltic Electronics Conference - BEC, Tallinn, October 6-8, 2014.

[85] J. Kõusaar, R. Ubar, I. Aleksejev. Complex Delay Fault Reasoning with Sequential 7-valued Algebra. Proceedings of 16th IEEE Latin American Test Symposium - LATS, Puerto Vallarta, Mexico, March 25-27, 2015.

[86] R. Ubar, J. Kõusaar, S. Devadze, J. Raik. Transition Delay Fault Simulation with Parallel Critical Path Back-Tracing and 7-valued Algebra. Microprocessors and Microsystems - MICPRO, Elsevier, Volume 39, Issue 8, Nov. 2015, pp. 1130-1138.

[87] R. Ubar, S. Devadze, J. Raik, A. Jutman. Parallel Fault Backtracing for Calculation of Fault Coverage. 13th ASPDAC, Seoul, Korea, 2008, pp. 667-672.

[88] F. Brglez, H. Fujiwara, A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran, in: Proc. of the International Test Conference, 1985, pp. 785-794.

[89] H.K. Lee, D.S. Ha, SOPRANO: an efficent automatic test pattern generator for stuck-open faults in CMOS combinational circuits, DAC, Orlando, FL, 1990, pp. 660-666.

[90] F. Brglez, D. Bryan, K. Kominski, Combinational profiles of sequential benchmark circuits, in: Int. Symp. on Circuits and Systems, 1989, pp. 1929-1934.

[91] F. Corno, M.S. Reorda, G. Squillero, RT-level ITC’99 benchmarks and first ATPG results, Proc. IEEE Des. Test Comput. 17 (3) (2000) 44-53.

[92] R. Ubar, J. Kõusaar, M. Gorev, S. Devadze. Combinational Fault Simulation in Sequential Circuits. Proceedings of International Symposium on Circuits and Systems - ISCAS, Lisbon, Portugal, 24-27 May, 2015, pp. 1-4.

[93] 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*, 2013.

[94] J. Kõusaar, R. Ubar, S. Kostin, S. Devadze, J. Raik. Parallel Critical Path Tracing Fault Simulation in Sequential Circuits. 25th International Conference Mixed Design of Integrated Circuits and Systems - MIXDES, June 21-23, 2018, Gdynia, Poland.

[95] J. Kõusaar, S. Kostin, R. Ubar, S. Devadze, J. Raik, “Exact Parallel Critical Path Fault Tracing to Speed-Up Fault Simulation in Sequential Circuits”, *International Journal of Microelectronics and Computer Science*, vol. 9, pp. 9-18, October 2018.

[96] R. Ubar, J. Raik, S. Kostin, J. Kõusaar. Multiple Fault Diagnosis with BDD based Boolean Differential Equations. Proc. of Baltic Electronics Conference, Tallinn, October 3-5, 2012.

[97] M. Gorev, R. Ubar, S. Devadze. 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, 9-13 March, 2015.

[98] M. Abramovici, M.A. Breuer, A.D. Friedman. “Digital Systems Testing and Testable Design”. IEEE Press, 1990.

[99] S. Kajihara, T. Sumioka, K. Kinoshita. “Test Generation for Multiple Faults Based on Parallel Vector Pair Analysis”. IEEE/ACM Int. Conf. on CAD – ICCAD’93, 1993, pp. 436-439.

[100] R. Ubar, J. Raik, A. Jutman, M. Jenihhin. “Diagnostic Modeling of Digital Systems with Multi-Level Decision Diagrams”. “Design and Test Technology for Dependable Systems-on-chip”, Raimund Ubar, Jaan Raik, Heinrich Theodor Vierhaus (Eds.), 2011, pp. 92-118.

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

[102] R. Ubar: Fault Diagnosis in Combinational Circuits by Solving Boolean Differential Equations, Automatics and Telemechanics, No. 11, Moscow, 1979, pp. 170-183 (in Russian).

[103] R. Ubar, J. Raik, A. Jutman, M. Jenihhin. “Diagnostic Modeling of Digital Systems with Multi-Level Decision Diagrams”. “Design and Test Technology for Dependable Systems-on-chip”, Raimund Ubar, Jaan Raik, Heinrich Theodor Vierhaus (Eds.), 2011, pp. 92-118.

Acknowledgements

I would like to express my gratitude towards people who believed in me despite the bumpy ride. Without you, this thesis would have never been finished.

Especially I would like to thank my supervisor, Professor Ubar who has believed in me during my PhD studies, although this journey was far from smooth sailing. Also, special thanks to Dr Maarja Kruusmaa, Dr Margus Kruus for understanding and giving me a second chance.

I would also like to thank Kadi, my children Jan and Simon. Thank you all for fortitude. My wife Maarja, for your calm attitude and continuous inspiration!

Furthermore, I would like to thank my colleagues from Warren.io and OYE Network LTD. for patience, while I was busy with my studies. Rauno, Sven-Erik, Kadri, Kaido, Avo, Marko, Henry and most of all, Tarmo. Thank you for your forbearance. Edukhan, thank you for helping me with graphics for this thesis!

Finally, I would like to thank my co-supervisor Professor Raik PhD, Dr. Siavoosh Payandeh Azad PhD, Dr. Maksim Gorev, Dr. Priit Ruberg, Karl Janson and Hardi Selg. Thank you for your continuous positive attitude and support. And Heljo Saar, who provided me guidance when my vision was blurred.

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), Estonian IT Foundation (EITSA) and Information Technology Foundation for Education (HITSA).

Thank you all!

Abstract
Methods for Improving the Accuracy and Efficiency of Fault Simulation in Digital Systems

In the era of emerging nanoelectronic technology, the ever-growing complexity of electronic designs requires increased design productivity and speed-up of the design and test tools to cope with strengthening requirements, both to cover a time to market and the quality of today's electronic products.

Logic-level simulation is one of the most critical processes in digital design for verification purposes, fault simulation, test generation and fault diagnosis. As speed of circuits is increasing constantly, a major factor affecting the product quality level have become violations of the performance specifications. This trend, in turn has triggered intensive research in the field of delay faults testing.

Another important topic is fault simulation in sequential circuits, which due to the presence of memory do not allow as fast fault simulation as in combinational circuits. The key challenge is to increase the speed of fault analysis in sequential circuits, to cope with the ever growing complexity of circuits. This has motivated to rethink the traditional methods and algorithms of fault simulation in sequential circuits, to discover and develop new ideas, making the existing concepts and approaches more efficient, thus, increasing the performance of fault simulation.

The efficiency and speed of simulation highly depends on the data structures used in simulation procedures and how these procedures are organized. Binary Decision Diagrams (BDD) has become state-of-the-art data structure in the VLSI CAD tools for representation and manipulation of Boolean Functions. As it was recently shown, a subclass of Structurally Synthesized BDDs (SSBDD), opens new possibilities to reduce the modeling complexity of digital circuits with preserving the accuracy of reflecting structural properties of circuits.

The findings above motivated to carry out research and investigations of using SSBDDs to introduce improvements into current approaches to fault simulation, especially, targeting delay fault simulation fault simulation in sequential circuits, and diagnostic fault simulation.

The general focus of the thesis is fault simulation in digital circuits. As the problem of fault simulation, in general, is very large, the research in Thesis is concentrated on extensions and developing new applications for the following existing simulation concepts:

- Extension of the class of delay faults in digital circuits
- Extension of the method of critical path tracing, as a very fast fault simulation concept used so far only for combinational circuits, for using also in sequential circuits.
- Extension of the field of using Structurally Synthesized BDDs (SSBDD), applied so far in fault simulation and test generation, for using also in fault diagnosis.

As an outcome, the following new results in the research field of fault simulation in digital circuits have been achieved:

- A new delay fault model (non-robust functionally sensitized TDF), and a new delay fault simulation method were developed, which allowed to improve the quality of delay fault testing.
- Two new approaches for fault simulation in sequential circuits were developed, based on applying parallel critical path fault back-tracing, used so far only for combinational circuits, which allowed considerable speed-up of fault simulation.
- A new type of BDDs called Diagnostic Structurally Synthesized BDD (DSSBDD) was introduced, and for the first time, a method was proposed for fault diagnosis in digital circuits in a general case of multiple fault assumption, by solving Boolean differential equations using DSSBDDs.

The results of the thesis are presented in 9 scientific works, which are published as three journal papers and six conference papers, majority of which are included into the IEEExplore digital library.

Lühikokkuvõte
Meetodid digitaalsüsteemide rikete simuleerimise täpsuse ja efektiivsuse tõstmiseks

Nanoelektroonika ajastu digitaalelektroonika disainide pidevalt kasvav keerukus on järsult suurendanud vajadust võimekamate projekteerimis- ning testimisvahendite järele, et sammu pidada üha tugevnevate nõuetega nii tootmistsükli kestuse kui toodete kvaliteedi osas .

Loogikataseme simulatsioon on digitaalelektroonika disainide verifitseerimisel, rikete simuleerimisel, testide genereerimisel ning rikkediagnoosil üks olulisemaid protsesse. Digielektroonika taktsageduse pidev kasv on põhjustanud jõudlust puudutavatele nõuetele mittevastavuse kujunemise peamiseks probleemiks tootekvaliteedi tagamise osas. See omakorda on kaasa aidanud teadustööde arvu plahvatuslikule kasvule viiterikete valdkonnas.

Mitte vähem olulisem valdkond diagnostikas on rikete simulatsioon järjestikskeemides, mis mälu kaasatuse tõttu skeemis jääb simulatsiooniprotsessi kiiruse poolest selgelt maha kombinatsiooniskeemidest. Sealjuures jääb peamiseks küsimuseks, kuidas kiirendada rikete analüüsi protsessi, võttes arvesse asjaolu et pidevalt suureneb ka skeemide keerukus. Selle küsimuse valguses on kasvanud arusaam, et rikete simulatsiooni tõhustamiseks tuleb senised meetodid ja algoritmid kriitiliselt üle vaadata ning suurelt osalt asendada uutel kontseptsioonidel põhinevatega.

Rikete simulatsiooni tõhusus ja kiirus sõltub suuresti simulatsiooniprotseduurides kasutatavatest andmestruktuuridest ning sellest, kuidas protseduurid korraldatakse. Binaarsed otsustusdiagrammid (BDD) on muutunud CAD - tööriistades prevaleerivateks andmestruktuurideks Boole'i funktsioonide esitamisel ja rakendamisel. Hiljutiste uurimuste kohaselt on näidatud struktuurselt sünteesitud BDD-de (SSBDD) kasutust kui uut efektiivset võimalust digitaalskeemide modelleerimise keerukuse vähendamisel, säilitades sealjuures kõik skeemide struktuursed omadused.

Ülaltoodud järeldused ajendasid SSBDD-de kasutamist uurima võimalust viia sisse täiendused hetkel kasutusel olevatesse rikkesimulatsiooni põhimõtetesse, seda eelkõige viiterikete simulatsioonis ning samuti ka järjestikskeemide diagnostilises rikete simulatsioonis.

Doktoritöö üldine fookus on rikete simulatsioon digitaalskeemides. Kuna rikete simulatsiooni probleemistik on suur, keskendub väitekirjas esitatu seniste meetodite täiendamisele ja uute rakenduste väljatöötamisele järgmiste olemasolevate simulatsioonikontseptsioonide jaoks:

- Täiendused viiterikete klassile digitaalskeemides.
- Kriitilise tee järgimise meetodi kui väga kiire rikete simulatsiooni kontseptsiooni laiendamine järjestikskeemidele, olles seni kasutusel vaid kombinatsiooniskeemide puhul.
- Struktuurselt sünteesitud BDD-de (SSBDD), mida on siiani rakendatud rikete simulatsioonil ja testide genereerimisel, kasutamine ka rikete diagnoosimisel.

Selle tulemusena on digitaalskeemide rikete simulatsiooni uurimisvaldkonnas saavutatud järgmised uuenduslikud tulemused:

- Välja on töötatud uus viiterikete mudel (non-robust functionally sensitized TDF) ja uus viiterikete simulatsioonimeetod, parandades seeläbi viiterikete testimise kvaliteeti.
- Välja on töötatud kaks uut lähenemist rikete simulatsiooniks järjestikskeemides, tuginedes paralleelsele kriitilisele tee tagasijärgimisele, mida seni on kasutatud ainult kombinatsiooniskeemide jaoks, võimaldades sellega rikete simulatsiooni kiiruse märkimisväärse kasvu.
- Tutvustatud on uut tüüpi BDD-d (diagnostiline struktuurselt sünteesitud BDD - DSSBDD), ning esmakordselt on välja pakutud meetod digitaalskeemides rikete diagnoosimiseks mitme rikke korraga eksisteerimise eelduse puhul, mis baseerub Boole´i diferentsiaalvõrrandite lahendamisel kasutades DSSBDD-d.

Väitekirja tulemused on esitatud 9 teadusartiklis, mis on avaldatud kolme ajakirja- ja kuue konverentsiartiklina. Antud artiklitest enamus sisalduvad IEEExplore'i digitaalses raamatukogus.

Appendix

Publication I
R. Ubar, J. Raik, S. Kostin, **J. Kõusaar**. Multiple Fault Diagnosis with BDD based Boolean Differential Equations. Proc. of Baltic Electronics Conference, Tallinn, October 3-5, 2012.

Multiple Fault Diagnosis with BDD based Boolean Differential Equations

Raimund Ubar, Jaan Raik, Sergei Kostin, Jaak Kõusaar

TTU, Ehitajate tee 5, 19086 Tallinn, Estonia,
E-mails: {raiub, jaan,skostin}@ati.ttu.ee, jaak.kousaar@gmail.com

ABSTRACT: We present a new idea for multiple fault diagnosis in combinational circuits, which combines the concept of multiple fault testing by test groups and solving Boolean differential equations by manipulation of BDDs. A discussion is presented how this approach can be used as a basis for hierarchical fault diagnosis to cope with the complexity problem.

1. Introduction

Stuck-at-fault (SAF) model is used to model defects in digital circuits. While single SAF (SSAF) model provides some coverage of logic defects, multiple SAF (MSAF) model provides comprehensively wider coverage for a variety of defects which cannot be modeled by SSAF. If a high defect coverage is desired it is obvious that to detect all logic faults, every detectable MSAF must be tested. Since a n-line circuit may have 3n – 1 faulty situations compared to 2n faulty situations under the SSAF model, it makes impossible to count all MSAF separately, and to use them as direct test generation objectives. On the other hand, to use SSAF as test objectives raises a problem to avoid fault masking because of possible MSAF [1].

Test pair concept is used to generate test patterns for MSAF by targeting SSAF as test generation objectives [2,3]. However, in [4,5] it was shown that not always test pairs can give a guarantee to avoid fault masking.

In [4,5], a conception of test groups was proposed. Differently from known approaches we don't target the faults themselves as the objectives of testing. Instead, the goal of testing will be to verify the correctness of selected subcircuit. A special case of a test group may be a test pair which has the goal to prove the correctness of a selected signal path in the circuit. A complete test is generated as a set of test groups, and the pass of each test group identifies a fault-free part of the circuit regardless the effects of all possible faults which might be present in other places of the circuit. The problems with test groups are twofold. First, they may not always exist, and second, if a test group will not pass, it would be difficult to diagnose the fault location.

In this paper, we present a method of combining the use of test groups with the method of fault diagnosis by solving Boolean differential equations whereas to find the solutions, Structurally Synthesized BDDs (SSBDD) are exploited.

The rest of the paper is organized as follows. Section 2 presents the method of fault diagnosis with Boolean differential equations. In Section 3 we show how the equations can be solved by manipulations of BDDs. Section 4 presents a discussion how the fault diagnosis by test groups can be supported by BDD based solving of Boolean differential equations to facilitate hierarchical fault diagnosis, Section 5 presents some experimental data, and Section 6 concludes the paper.

2. Modeling fault diagnosis with Boolean differential equations

Consider a single output combinational circuit with n inputs as a Boolean function $y = F(x_1, x_2, \dots, x_n)$. A test experiment with the circuit can be modeled as a Boolean full diferential [6]

$$dy = y \oplus F((x_1 \oplus dx_1), (x_2 \oplus dx_2), \dots, (x_n \oplus dx_n)). \quad (1)$$

When applying a test pattern T_t, the diagnosis on the basis of this experiment can be represented as a logic assertion

$$dy^t = F^t(dx_1^t, dx_2^t, \dots, dx_n^t) \quad (2)$$

where

$$dx_j^t \in \{dx_j, \overline{dx}_j\},$$

$dy^t \in \{0,1\}$ denotes the test result: $dy^t = 0$, if the test pattern T_t has passed, and $dy^t = 1$, if the test pattern T_t has failed, dx_j means that there is a suspected fault related to x_j, and $\overline{dx}_j$ means that no fault at x_j is suspected. The fault type under question is defined by the value of x_j at the given pattern. To create the possibility of manipulations with faults of different types, let us introduce for each variable x_j a set of suspected diagnostic states

$$DV(x_j) = \{dx_j^0, \overline{dx}_j^0, dx_j^1, \overline{dx}_j^1, \overline{dx}_j\}$$

where the assertions $dx_j^0, \overline{dx}_j^0, dx_j^1, \overline{dx}_j^1, \overline{dx}_j$ which may be true and false have the following meanings: the fault $x_j \equiv 1$ is suspected (the upper index at dx_j means the value of x_j at the given test pattern), the fault $x_j \equiv 1$ is not supected, the fault $x_j \equiv 0$ is suspected, the fault $x_j \equiv 0$ is not suspected, and no faults are suspected at the variable x_j, respectively.

Let us introduce on the basis of (1) the following diagnostic equation as a true assertion

$$D(T_t) = \overline{dy^t} \oplus (y^t \oplus F^t) = 1 \quad (3)$$

Assume, we have carried out a test experiment $T = (T_1,T_2)$ with two test patterns, and we have a got the following test results (dy^1,dy^2), respectively. The statement about fault diagnosis $D(T)$ based on the test T can be calculated from the logic multiplication of two assertions

$$D(T) = D(T_1) \wedge D(T_2) = 1. \qquad (4)$$

For processing the diagnosis equations (4), we can use the 5-valued algebra depicted in Table 1, to find out the inconsistencies of two assertions and to carry out all the possible simplifications in (4).

Table 1. *5-valued algebra for calculating Boolean differentials*

	dx^0	$\overline{dx^0}$	dx^1	$\overline{dx^1}$	$\overline{dx}$
dx^0	dx^0	∅	∅	dx^0	∅
$\overline{dx^0}$	∅	$\overline{dx^0}$	dx^1	$\overline{dx}$	$\overline{dx}$
dx^1	∅	dx^1	dx^1	∅	∅
$\overline{dx^1}$	dx^0	$\overline{dx}$	∅	$\overline{dx^1}$	$\overline{dx}$
$\overline{dx}$	∅	$\overline{dx}$	∅	$\overline{dx}$	$\overline{dx}$

If the final reduced assertion $D(T)$ will consist of a single DNF term, the diagnosis statement is unambiguous. More than one terms will mean ambiguity. The more test patterns we will use in the test experiment, the less ambiguous the diagnosis will become.

Example 1. Consider a circuit in Fig.1, and the test experiment with the diagnosis after each test pattern in Table 2.

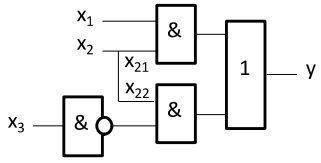


Fig. 1. *Combinational circuit*

The Table 2 illustrates the course of the diagnostic process in case where all test patterns pass. After applying the first two test patterns, we can state unambiguously that the signal path from the input x_1 up to the output y is working correctly, and the fault $x_{21} \equiv 0$ is missing. After the third test, we know that the fault $x_{21} \equiv 1$ is as well missing. After the 5th test we can state that the circuit is functioning correctly.

3. Modeling fault diagnosis with BDDs

BDDs have been proved to be an efficient tool for manipulations with Boolean functions [7]. Since the diagnostic equations (3) represent Boolean expressions, and the diagnostic processing of test results according to (4), we can easily find the final diagnostic assertions by manipulations with BDDs, which allows us to avoid the explosion of the expressions (4).

Table 2. *Diagnostic process with 5 passed test patterns*

T_t	x_1	x_2	x_3	y	Diagnostic assertions
T_1	0	1	1	0	$(\overline{dx_1^0} \vee dx_{21}^1)(dx_{22}^1 \vee \overline{dx_3^1}) = 1$
T_2	1	1	1	1	$\overline{dx_1^1}\,\overline{dx_{21}^1} \vee \overline{dx_{22}^1}\,dx_3^1 = 1$
$D_2=D(T_1,T_2)$					$\overline{dx_1}\,\overline{dx_{21}^1}(dx_{22}^1 \vee \overline{dx_3^1}) = 1$
T_3	1	0	1	0	$(dx_1^1 \vee \overline{dx_{21}^0})(\overline{dx}_{22}^0 \vee \overline{dx_3^1}) = 1$
$D_3=D(T_1,T_2,T_3)$					$\overline{dx_1}\,\overline{dx_{21}}(\overline{dx_3^1} \vee \overline{dx_{22}^0}dx_{22}^1) = 1$
T_4	0	1	0	1	$dx_1^0\,\overline{dx_{21}^1} \vee \overline{dx_{22}^1}\,\overline{dx_3^0} = 1$
$D_4=D(T_1,T_2,T_3,T_4)$					$\overline{dx_1}\,\overline{dx_{21}}\,\overline{dx_3}\,\overline{dx_{22}^1} = 1$
T_5	0	0	0	0	$(\overline{dx_1^0} \vee \overline{dx_{21}^0})(\overline{dx_{22}^0} \vee dx_3^0) = 1$
$D_5=D(T_1,T_2,T_3,T_4,T_5)$					$\overline{dx_1}\,\overline{dx_{21}}\,\overline{dx_{22}}\,\overline{dx_3} = 1$

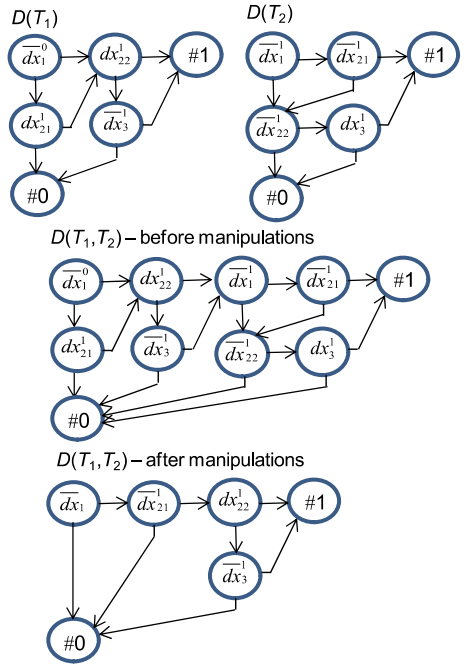


Fig.3. *SSBDDs for diagnostic experiment $D(T_1,T_2)$*

Algorithm. Consider a diagnostic experiment based on the test set $T = \{T_1,T_2,...,T_n\}$. For modeling the faults in the expressions $D(T_k)$, $T_k \in T$, we will use Structurally Synthesized BDDs (SSBDD) [8]. The BDD for $D(T_1)$ will represent the first current diagnosis D_1. Iteratively, each next diagnosis D_k after the experiment with test T_k will

be calculated as $D_k = D_{k-1} \wedge D(T_k)$. First, the joint BDD will be created for D_k, where each 1-path in the BDD [8] corresponds to a term in the DNF of D_k represented by BDD. Next, all the 1-paths in the BDD will be processed, and the inconsistent paths, in accordance to the 5-valued algebra in Table 1, will be excluded from the BDD. The final simplified BDD for $D(T_k)$ represents the statement of fault diagnosis derived from the given test experiment.

Example 2. Consider in Fig.2 the two SSBDDs for representing $D(T_1)$ and $D(T_2)$ in Table 2 (in the first two rows). The labels on the edges of the BDDs are omitted, the right-hand edge from a node corresponds to the value 1, and the down-hand edge corresponds to the value 0 of the node variable. The graph for $D(T_1)$ contains four 1-paths, the graph for $D(T_2)$ two, and the merged BDD for $D(T_1,T_2)$ contains eight 1-paths. By processing the paths we exclude 6 inconsistent ones, and create a BDD with two 1-paths, which represents a statement about two possible diagnostic cases $\overline{dx_1}\,dx_{21}^1 dx_{22}^1$ $\overline{dx_1}\,dx_{21}^1 dx_3^1$

4. Test groups and hierarchical fault diagnosis

In [4,5] a conception of test groups was introduced, and the necessary conditions for detecting MSAF in combinational circuits were introduced. The goal of a test group is to verify the correctness of a selected part of the circuit. In case of passing of all the test groups, the circuit is proven fault-free. In case when not all test groups will pass, a fault diagnosis is needed, which can be carried out by solving diagnostic equations locally, i.e. in the selected region of the circuit targeted by the test group.

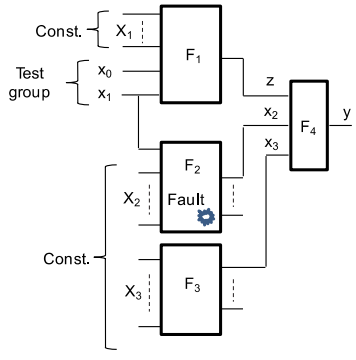


Fig.3. *Hierarchical fault diagnosis*

Consider a circuit in Fig.3 as a higher level network with blocks F_1, F_2, F_3, and F_4. Let us start the fault diagnosis first, at the lower level, by selecting a subcircuit with inputs x_0 and x_1 as a target in the block F_1. The subcircuit will be tested along an activated test path L through the wire z up to the output y of the circuit.

Assume, we have a test group $TG = \{T_0, T_1, T_2\}$ applying to the inputs x_0 and x_1 a set of changing patterns whereas the values on all other primary inputs of the circuit are kept, according to the test group conception [4,5], constant. The role of the test group is to activate for all the test patterns T_0, T_1, T_2 a joint signal path L. If the test group will pass, we conclude that the path L is working correctly. Each additional passed test group will add information about other correct signal paths. This information will help to locate the faults if some test groups will fail.

If we have proven the correctness of a subset S of signal paths then we can use these paths for sending correct signals to the needed connections in the high network level which makes easier to test other blocks at the lower level. For example, if we can force correct signals on the wires z and x_3 in Fig.3, to activate a test path from x_2 up to the output y, we may carry out the fault reasoning in the block F_2 only locally to reduce the complexity of the diagnosis problem

Example 3. Consider the circuit in Fig.3, and apply the test group $TG = \{T_0, T_1, T_2\}$ to the block F_1, so that the inputs x_0 and x_1 will have the local patterns {11,01,10} and the values on all other primary inputs of the circuit are kept constant. Assume that the activated by TG test path can be represented by a function

$$y = F(x_0, x_1, x_2, x_3, X_1). \qquad (5)$$

Assume also that all the signals activating the test path of the test group and originating at the inputs X_1 have been proved as correct. This allows to reduce the diagnosis problem raised by the test group TG to processing of the simplified function

$$y = x_0 x_1 \vee x_2 \vee x_3.$$

Table 3. *Diagnostic processes for the circuit in Fig.3*

T_t	x_0	x_1	x_2	x_3	y	dy	Assertions
T_1	1	1	0	0	1	0	$\overline{dx_0}^1 \overline{dx_1}^1 \vee dx_2^0 \vee dx_3^0$
T_2	0	1	0	0	0	0	$(\overline{dx_0}^0 \vee dx_1^1)\overline{dx_2}^0 \overline{dx_3}^0$
$D(T_1,T_2)$							$\overline{dx_0}\,\overline{dx_1}^1 \overline{dx_2}^0 \overline{dx_3}^0$
T_3	1	0	0	0	0	0	$(dx_0^1 \vee \overline{dx_1}^0)\overline{dx_2}^0 \overline{dx_3}^0$
$D(T_1,T_2,T_3)$							$\overline{dx_0}\,\overline{dx_1}\,\overline{dx_2}^0 \overline{dx_3}^0$
T_3'	1	0	0	0	0	1	$\overline{dx_0}^1 dx_1^0 \vee dx_2^{0*} \vee dx_3^0$
$D(T_1,T_2,T_3')$							$\overline{dx_0}\,\overline{dx_1}^1 \overline{dx_3}^0 (dx_1^0 \vee dx_2^{0*} \overline{dx_2}^0)$

The diagnostic process with three passed test patterns of the test group is depicted in Table 3. The final assertion $D(T_0,T_1,T_2)$ states that no faults are present along the signal paths from the inputs x_0 and x_1 up to the output y, and the faults $x_2 \equiv 1$ and $x_3 \equiv 1$ are missing on the inputs of the block F_4.

The knowledge about the correctness of the wire z and the missing fault $x_3 \equiv 1$ allows to carry out the fault reasoning in the block F_2 only locally.

Suppose the pattern $T_3' \in TG$ will fail. The diagnostic statement $D(T_1,T_2,T_3')$ refers to either the fault candidate $x_1 \equiv 1$ or to the unstabile behaviour of the wire x_2 during execution of the test group. The reason of the unstability of x_2 may be a fault in the block F_2 which because of the changing value of x_1 may sometimes influence on $x2$, and sometimes not. In this case the test group has not fulfilled its role to prove the correctness of the wire z.

The role of the Boolean full differential as shown in Example 3, lays on specifying the fault candidates in the case when the test group will fail. The method of solving differential equations will help also in this case when some of the test groups cannot be synthesized and it would not be possible to prove the correctness of some parts of the circuit.

5. Experimental data

In Table 4, experimental data are presented regarding the test group synthesis for the ISCAS'85 benchmark circuits [9]. The columns 3 and 4 show the number of patterns in the SSAF test and in the test groups, respectively. The 3-pattern test groups were built for the gates of the circuits whereas many test groups were possible to merge. Repeated patterns were removed from the test set. Still, the synthesis of test groups resulted in the test sets many times larger compared to the traditional SSAF tests. The fair comparison between the SSAF and MSAF test lengths, however, cannot be done in the present research, since the test groups for different outputs were not merged. Merging of test groups will provide a significant reduction of the MSAF test length.

Table 4. *Experimental data of generating tgest groups for multiple fault diagnosis*

Circ.	Gates #	SSAF test #	MSAF test #	Group cover %
c432	275	53	314	82,91
c499	683	86	482	67,2
c880	429	84	546	99,8
c1355	579	86	514	65,6
c1908	776	123	621	96,3
c2670	1192	103	820	76,3
c3540	1514	148	995	80,3
c5315	2240	104	1523	91,8
c6288	2480	22	465	98,1
c7552	3163	202	1863	87,8

The group coverage means the persentage of the test groups that were built successfully. The test group coverage characterizes the feasibility of the concept to prove the correctness of subcircuits instead of targeting the faults to be tested. For diagnosing the subcircuits not covered by test groups, the tool of Boolean full differential can be used.

6. Conclusions

In this paper we investigated the two sides of the fault diagnosis problem: how to develop efficient diagnostic test sequences and how to locate the faults if some test patterns will not pass. The test group concept affords to concentrate on the diagnosis of small parts of circuits to cope with the complexity problem by hierarchical fault reasoning. On the other hand, using test groups allows to prove the correctnesses of selected parts in a given circuit.

We present a new idea for multiple fault diagnosis, which combines the concept of multiple fault testing by test groups with solving Boolean differential equations by manipulation of BDDs. The role of the test groups is to prove step by step the correctnesses of larger and larger parts of the given circuit. If the test groups cover the whole circuit, and all the test groups will pass, the fault reasoning is not needed. If some parts of the circuit cannot be excercized by test groups, the fault reasoning by solving Boolean differential equations is needed to cope with the MSAF case. As the experiments showed, it was not possible to synthesize in average 15% of 3-pattern test groups for the gates. The tool of Boolean full differential is useful for reasoning MSAF cases also when some of the test groups will not pass.

To avoid the memory explosion when solving the Boolean differential equations, the manipulation of BDDs based on the proposed 5-valued algebra for fault reasoning will serve as an efficient tool.

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] M.Abramovici, M.A.Breuer, A.D.Friedman. Digital Systems Testing and Testable Design. IEEE Press, 1990.

[2] H.Cox, J.Rajski. A Method of Fault Analysis for Test Generation and Fault Diagnosis. IEEE Trans. on CAD, vol.7, no.7, 1988, pp.813-833.

[3] S.Kajihara, T.Sumioka, K.Kinoshita. Test Generation for Multiple Faults Based on Parallel Vector Pair Analysis. IEEE/ACM Int. Conf. on CAD – ICCAD'93, 1993, pp. 436-439.

[4] R.Ubar, S.Kostin, J.Raik. About Robustness of Test Patterns Regarding Multiple Faults. 13th IEEE Latin American Test Workshop, Quito, Ecuador, April 10-13, 2012.

[5] R.Ubar, S.Kostin, J.Raik. Multiple Stuck-at-Fault Detection Theorem. The 15th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, Tallinn, Estonia, April 18-20, 2012.

[6] A.Thayse, M.Davio. Boolean Differential Calculus and its Applications to Switching Theory. IEEE Trans. Comput. V.C-22, No.4, pp.409.4, pp.409.420, 1973.

[7] Bryant, R.E. (1986). Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Computers (Vol.C-35), No 8, pp.667-690.

[8] R.Ubar, J.Raik, A.Jutman, M.Jenihhin. Diagnostic Modeling of Digital Systems with Multi-Level Decision Diagrams. Design and Test Technology for Dependable Systems-on-chip, Raimund Ubar, Jaan Raik, Heinrich Theodor Vierhaus (Eds.), 2011, pp.92-118.

[9] F.Brglez, H.Fujiwara. A Neutral Netlist of 10 Combinational

Publication III

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, 2014.

Critical Path Tracing based Simulation of Transition Delay Faults

Jaak Kõusaar, Raimund Ubar, Sergei Devadze, Jaan Raik
Department of Computer Engineering
Tallinn University of Technology, Tallinn, Estonia
jaak.kousaar@gmail.com, {raiub, serega, jaan}@pld.ttu.ee

***Abstract*—A new method is presented for simulating of transition delay faults (TDF). The main idea of the method is to extend the TDF model, traditionally considered as a class of robustly tested delay faults, to a class of TDFs with extended detection conditions. Three known fault classes of delay fault sensitization are considered: robust, non-robust and functional sensitization of delay faults. Additionally, a new fault class is introduced, called non-robust functionally sensitized delay fault. A novel fault analysis algorithm based on 7-valued algebra is presented, which delivers the fault coverage for all mentioned four types of TDFs.**

Keywords – transition delay faults, non-robust and functional sensitization, critical path fault tracing, 7-valued algebra

1 Introduction

The purpose of delay testing is to ascertain that manufactured digital circuits meet their timing specifications. Delay faults can be modeled in different ways, among which the most common models are transition delay fault (TDF) model [1], and path delay fault (PDF) model [2]. The TDF model captures large delay defects that affect single locations in the circuit whereas the PDF model captures small delays, such that each one by itself may not cause the circuit to fail, but their cumulative effect along a path from inputs to outputs may result in faulty behavior. The TDF is as well used as a logic model for stuck-open faults in CMOS circuits, which either suppress or delay the occurrence of certain transitions [3].

The main advantage of the TDF model is the linearity of the number of faults in terms of the number of connections in gate-level circuits. Another advantage is that the stuck-at-fault (SAF) test generation and fault simulation tools can be directly used for handling TDFs. The disadvantage of TDFs is that the expectation that the delay fault in a single location is large enough for propagating up to the observation points might not be realistic. On the other hand, the disadvantages of the PDF model is that the number of paths in circuits is huge, the number of robustly testable paths is in general very small, and as shown in [4], the non-robust tests may not detect delays in certain situations where the cumulative effect of small delays would be sufficient to be tested non-robustly.

In [4,5] a novel *Transition Path Delay Fault* (TPDF) model was proposed where the ideas of TDF and PDF models are combined to improve the capability of TDFs to detect the delays by combining tests for TDFs along the target paths.

In [6] a new TDF model is proposed called *As Late As Possible Transition Fault* (ALAPTF). The model aims at detecting cumulatively smaller delays, which will be missed by both the traditional TDF and the PDF models. The model makes sure that each transition is launched as late as possible at the TDF site, accumulating the small delay defects along its way.

Both approaches [4] and [6] are based on the idea of creating paths along which the TDFs are consecutively tested.

In both approaches we need to have continuous activated paths or segments of paths along which the TDF will robustly propagate. In general, such paths may be missing, or it may be difficult to create paths along which all the TDFs are detected. In these cases, it may be reasonable to alleviate the constraints on consecutive propagation of TDFs robustly along the tested path, so that in some gates or segments the TDFs may propagate non-robustly or along discontinuous paths.

In [4], it was shown that a test for TPDF corresponds to a type of strong non-robust test [7] for a conventional PDF associated with the same path, however, with the difference that the TPDF test additionally detects the TDFs on every line of the path.

On the other hand, a strong non-robust test satisfies the weak non-robust propagation conditions [7], but not the opposite. As shown in [8], this results in a large gap in the number of detectable faults between TPDFs as defined in [4] and conventional PDFs when the weak non-robust propagation conditions are used for them. To bridge this gap, in [8], the detection conditions for TPDFs used in [4] were extended with hazard-based detection conditions for transition faults defined in [9]. When the hazard-based detection conditions are allowed for the transition faults included in a TPDF, it is not necessary to create transitions on every line of the path, rather it is also possible for lines on the path to have hazards. When some of the TDFs associated with a TPDF are detected under the hazard-based detection conditions, the resulting test will be a type of weak non-robust test.

In this paper, we further alleviate the conditions of detecting TDFs in addition to the hazard-based conditions (let us call them non-robust detection conditions for TDFs). We introduce into the detection conditions for TDFs so called *functional sensitization* [7] to handle multiple TDFs and a new type of conditions that we call *non-robust functional*

sensitization conditions which allows multiple TDF detection in hazard-based conditions. To determine the detectability of TDFs under mentioned conditions, we introduce 7-valued algebra which will support SAF simulation, and allows to classify all the detected TDFs into the classes of robustly detected, non-robustly detected, detected under functional sensitization conditions or under non-robust functional sensitization conditions.

We carry out the TDF simulation in two phases: traditional SAF simulation and additional check if the conditions needed for TDF detection are satisfied.

SAF simulation can be carried out in different ways: by parallel pattern single fault propagation [10], critical path tracing [11], deductive [12], or concurrent [13] fault simulation. The last three approaches are based on logic reasoning rather than simulation, and hence, are very powerful since all the detectable faults are calculated by a single run of the current test pattern. What they cannot do is to produce reasoning for many test patterns in parallel.

The Critical Path Tracing (CPT) approach consists of simulating the fault-free circuit and using the computed signal values for back-tracing all sensitized paths from primary outputs to primary inputs, to determine all the faults detectable by the given test pattern. The back-trace continues until the paths become non-sensitive or end at network primary inputs. Faults on the sensitive (critical) paths are detectable by the test. In the conventional CPT approach the detectable faults are found by a single run, however the precise results are only guaranteed in fan-out free circuits.

A modified critical path tracing technique that is linear time, exact, and complete is proposed in [14]. However, the rule based strategy used in this approach does not allow simultaneous parallel analysis of many patterns. Parallel critical path backward tracing in fan-out free regions combined with parallel simulation of stem faults was investigated in [15]. In [16], the concept of exact parallel critical path tracing was generalized for SAF cover analysis beyond the fan-out free regions, and in [17], the method was extended for simulation of a large fault class called X –fault.

In this paper we adjust the method proposed in [17] for parallel critical path tracing of TDFs, and we extend the detection conditions for TDFs compared to that proposed in [8,9]. Both phases of TDF simulation will be carried out by critical path backward tracing, the phase of SAF analysis is carried out in parallel for many patterns simultaneously, and the TDF type analysis is carried out pattern by pattern using the 7-valued algebra developed in the paper.

The organization of the paper is as follows. First, in Section 2, the concept of parallel critical path tracing method for calculating the detection of SAFs is described, and in Section 3, a novel 7-valued algebra for determining the types of detected TDFs is presented. Section 4 presents experimental results, and Section 5 concludes the paper.

2 SAF SIMULATION WITH CRITICAL PATH TRACING

In the first phase of TDF simulation we determine the detected SAF faults in the circuit for the given set of test patterns. For this purpose we use the parallel backward critical path tracing [16, 17]. In the following, the method is shortly described.

Consider a Fan-out-Free Region (FFR) represented by a Boolean function $y = F(x_1, \ldots, x_i, x_j, \ldots x_n)$. The task of SAF simulation can be referred to calculation of Boolean derivatives: if $\partial y/\partial x_j = 1$ then the fault is propagated from x_j to y. This check can be performed in parallel for a set of test patterns, so that each bit in a computer word corresponds to one test pattern. In order to extend the parallel critical path tracing beyond the fan-out free regions we use the concept of partial Boolean differentials [18].

Consider a gate (or an FFR) F, where a group of paths are re-converging from the same node x, as shown in Fig.1. Denote the Boolean function that represents the gate F as $y = F(x_1, \ldots, x_i, x_j, \ldots x_n)$.

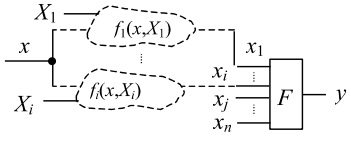


Fig.1. Reconvergent FFR in a circuit

Assume that the input variables $x_1,\ldots,x_i$ of the sub-circuit F are connected to the fan-out stem x via other sub-circuits represented by Boolean functions

$$x_1 = f_1(x,X_1), \ \ldots, \ x_i = f_i\,(x,X_i),$$

where the vectors X_i denote the nodes in the circuit which do not have paths to x. Then all possible fault propagation conditions from x to y can be described by full Boolean differential [18]:

$$dy = dF = y \oplus F((x_1 \oplus dx_1),\ldots,(x_i \oplus dx_i), (x_j \oplus dx_j),\ldots,(x_n \oplus dx_n)) \qquad (1)$$

where the Boolean variables dx and dy denote the erroneous changes of the values of x and y, respectively, caused by a propagated fault. In [16] we have shown that if a SAF at y is propagated and detected by a test pattern at primary outputs then the fault at the fan-out stem x is as well detected if

$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}),\ldots,(x_i \oplus \frac{\partial x_i}{\partial x}), x_j,\ldots,x_n) = 1 \cdot \qquad (2)$$

From the differential equation (2), a method results which generalizes the parallel exact critical path tracing beyond FFRs. As all the operations in (2) are Boolean, they can be carried out in parallel.

In a general case of nested re-convergences, the formula (2) can be used recursively [16,17]. If a SAF is detected by a test pattern on the output y of a sub-circuit in Fig.2 with two nested re-convergences, $y = F_y(x_1,z,X_y)$ and $z = F_z(x,X_z)$,

where X_y and X_z are not depending on x, then the fault at the common re-convergent fan-out stem is also detected if

$$d_x y = y \oplus F_y(x_1 \oplus \frac{\partial x_1}{\partial x}, z \oplus d_x F_z, X_y) = 1 \qquad (3)$$

.

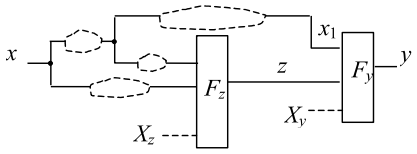


Fig.2. Nested re-convergences in a circuit

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 re-convergent FFR by consecutive calculation of the partial Boolean differentials, first $d_x F_z$, and then $d_x y$. The formula (3) can be iteratively generalized and constructed for any configuration of nested re-convergences by topological reasoning of the circuit. On the other hand, the derived full Boolean differentials can be easily transformed into fast computable critical path tracing procedures to be carried out in parallel for sets of test patterns. The details of the method can be found in [16,17].

As the result of the described parallel critical path tracing procedure, the following two tables are created:

(1) Fault table $FT = \| r_{i,j} \|$, $r_{i,j} \in \{0,1,\varnothing\}$ where i and j denote the numbers of simulated test patterns and nodes in the circuit, respectively; $r_{i,j} = 0$ (1) if the fault stuck-at 0(1) on the node j is detected by the pattern i, and $r_{i,j} = \varnothing$ if none of these faults is detected on the node j by the pattern i.

(2) Test pattern table $TP = \| p_{i,j} \|$ where $p_{i,j} \in \{0,1\}$ is the signal value on the node j produced by the pattern i.

The data of tables FT and TP will be used for mapping SAF faults from FT to TDF candidates to be confirmed during the second phase of TDF simulation.

3. Modeling of Transition Delay Faults

To detect a TDF, we have to apply two input patterns, $V = (V_1, V_2)$, so that V_1 will initialize the circuit, while V_2 will activate the fault and propagate its effect to some primary output. In the first phase of SAF simulation, we determine all the nodes in the circuit which were tested for SAF. For example, if the node x_i was tested for SAF $x_i \equiv 1$, then this node is the candidate for detecting the TDF "slow-to-fall" transition at this node, and if the node x_i was tested for SAF $x_i \equiv 0$, then this node is the candidate for detecting the TDF "slow-to-rise" transition at this node.

In conventional TDF testing it is needed that a test pair $V = (V_1, V_2)$ will assign to the node x_i different values, to initiate a transition, either $0 \rightarrow 1$ for detecting the TDF "slow-to rise", or $1 \rightarrow 0$ for detecting the TDF "slow-to-fall". These conditions we classify as conditions to detect the TDF robustly. To determine, if the TDF fault $v \rightarrow v'$ is detected on the line x_i, where $v \in \{0,1\}$, and v' means inverted v, it is needed that the test pattern V_1 will assign $x_i = v$, the test pattern V_2 will assign $x_i = v'$, and the SAF fault $x_i \equiv v$ is detected by V_2 at any primary output of the circuit.

In general case, e.g. when using the hazard-based detection conditions for the TDF $v \rightarrow v'$ on the node x_i, it is allowed to have a hazard or a pulse on the line x_i, under which x_i makes the transitions $v' \rightarrow v \rightarrow v'$ during the second pattern V_2 of the test. The fault is assumed to delay the $v \rightarrow v'$ transition of the pulse. Thus, in the presence of fault, x_i is assumed to make the transition $v' \rightarrow v$, and stay at v. So, the fault can be detected by a test that creates the pulse on x_i and propagates the pulse under the second test pattern to an output. In other words, at the hazard-based conditions it is not necessary to create a transition on every line of the path to detect a TDF. It is also possible for lines on the path to have hazards. The TPDFs detected under hazard-based conditions correspond to a type of weak-non-robust test. We call this case as a non-robustly tested TDF.

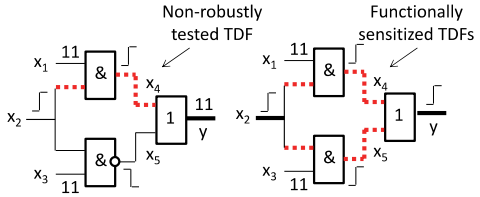


Fig.3. TDF is propagated along discontinuous paths

Consider an example in Fig.3 where a test pair $V = (V_1, V_2)$ is applied on the inputs with transition $0 \rightarrow 1$ on x_2. Assume that on both of the circuits the SAF faults at the outputs y are detected by V_2. In both circuits, the transition on the node x_2 is propagating further through two paths which re-converge at the node y.

In the case of Fig.3a, the fault effect on x_2 disappears, and does not reach the output y, because of different inversion parities on the re-converging paths. This is the reason why SAF $x_2 \equiv 0$ is not detected under V_2. Consequently, the TDF $0 \rightarrow 1$ on x_2 is not propagating to y, and the TPDF on the path x_2 - x_4 - y is not detected as well. However, the TDF $0 \rightarrow 1$ on x_4 can propagate non-robustly to y, and hence detected non-robustly. As the result, the TPDF on the path x_2 - x_4 - y can be detected non-robustly as well.

In Fig.3b, the TDF $0 \rightarrow 1$ on x_2 propagates to y, and is detected. However, the TDFs on x_4 and x_5 are not detected, because the SAFs $x_4 \equiv 0$ and $x_5 \equiv 0$ are not propagated under V_2. This means that there is no path in the circuit, where the TDFs are detected on all lines of the path, and consequently no TPDFs are detected in the circuit under test pair V. On the other hand, the TDFs at the nodes x_4 and x_5 are sensitized functionally [7], and can propagate to y as a multiple delay fault. This means that on both paths from x_2 to y, the TDFs are

detected on all lines under alleviated conditions (under functional sensitization conditions on the lines x_4 and x_5), and hence, the TPDFs for the paths from x_2 to y are as well detected under functional sensitization conditions.

To define the testing possibilities for TDFs, and alleviate in the same time the constraints on detection of TDFs, we introduce the following types of TDFs: *robustly tested* TDF, i.e. the conventional case (denote it as R-type), *non-robustly tested* TDF (N-type), *functionally sensitized* TDFs [7], i.e. the multiple delay fault case (F-type), and a new type of *non-robustly functionally sensitized* TDFs (X-type).

These types of testing of TDFs are illustrated in Fig.4. For all the cases depicted, it is assumed that the SAF fault on the output of the gate is detected under the test vector V_2.

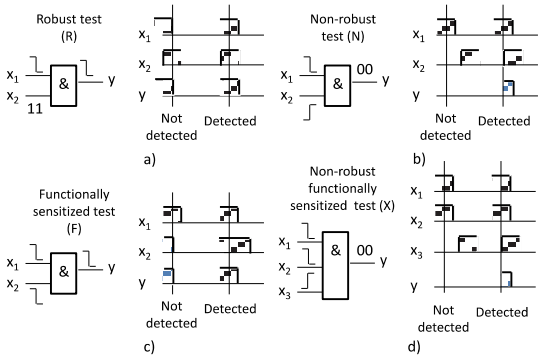


Fig.4. Four types of conditional detecting of TDFs

In all the figures in Fig.4 the TDF on the input x_1 is under test. The first diagram shows the waveforms of signals for the case where the delay fault on x_1 is missing or is not detected, and the second diagram shows the case where the delay fault is detected. In Fig.4a, the TDF on x_1 is tested robustly (conventional case, or TDF of type R). Fig.4b shows the case of non-robust testing (the TDF of type N). The TDF on x_1 may not be detected if the transition $0 \rightarrow 1$ on x_2 is late. If this transition will be not delayed, or is delayed less than the transition on x_1, the TDF will be detected. In Fig.4c, the TDF on x_1 is tested under functional sensitization conditions (the TDF of type F) [7]. This is the case when only multiple TDFs can be detected, i.e. the delay of the transition $1 \rightarrow 0$ on x_1 can be detected only if the same transition on x_2 is as well delayed. In the conventional meaning of TDF testing, the TDFs on the lines x_1 and x_2 can be regarded as untestable faults. Finally, in Fig.4d, the combination of the cases in Fig.4b and Fig.4c is considered (the TDF of type X). This is a new type of TDFs introduced in the paper. The multiple TDFs on x_1 and x_2 may not be detected if the transition $0 \rightarrow 1$ on x_3 is late. If this transition will be not delayed, or is delayed less than the both transitions on x_1 and x_2, the TDF will be detected.

The TPDF model [4] for a path P requires that all the TDFs on the path P were detected. The detection of the TDF types like N, F and X allows to alleviate the conditions under which the TDF will be considered as detected. However, the TDF detection conditions defined by N, F and X types are not guaranteed to happen. The goal of alleviation of the TDF detection conditions is to bridge the gap between the numbers of detectable TPDFs [4], and conventional PDFs, in a similar way as it was proposed in [8] by introducing the hazard-based TDF detection conditions (the type N of TDFs).

4 7-VALUED ALGEBRA FOR TDF SIMULATION

For simulating of TDFs and qualifying the types of the TDFs in accordance with the conditions they can be detected, we have developed two 7-valued algebras A1 and A2 as shown in Fig.5 for the gates OR/NOR. The similar algebras for the gates AND/NAND are dual. The algebras shown in Fig.5 are developed for two-input gates. If the number of inputs of the gate is bigger than two, then we have to represent the multi-input gate by an equivalent sequence of two-input gates. The algebras are defined by the set of values (symbols) $\{0,1,\varepsilon, h, N, F, X\}$, with interpretations explained in Fig.5, and by the binary calculation operators in a form of tables A1 and A2 in Fig.5.

The symbols 0, 1, ε, and h denote the signals on the inputs of the gate under analysis assigned by the test vector $V = (V_1, V_2)$. The symbols 0, 1 represent the steady signals 00 and 11, respectively, and the symbols ε and h represent the transitions 01 and 10, respectively. The symbols R, N, F and X define the type of the conditions under which the TDFs can be detected. The symbol $\varnothing$ means that the TDF on the gate input under analysis is not tested. The interpretation of R, N, F, and X is explained in Fig.4.

We put the fault types in the following dominance relation R > N, R > F, R > X, N > X, F > X. If the same TDF is detected under different types with different test patterns, always the dominating type should be assigned to the TDF. The highest dominating TDF type is R, and the lowest is X.

Interpretation of symbolic values	
0	00/No test
1	11/No test
h	10 /Robust
ε	01 / Robust
F	Functional delay
N	Non-robust delay
X	Non-robust functional delay
R	Robust delay
∅	N.A.

A1: Delay type calculation

OR	0	1	ε	h	N	F	X
0	0	1	ε	h	N	F	X
1	1	1	1	1	1	1	1
ε	ε	1	F	N	X	F	X
h	h	1	N	h	N	X	X

A2: Delay value calculation

OR	Calculated types						
	0	1	ε	h	N	F	X
0	∅	∅	∅	∅	∅	∅	∅
h	R	∅	∅	R	∅	∅	∅
ε	R	∅	F	N	N	F	X
1	∅	∅	∅	∅	∅	∅	∅

Fig.5. 7-valued algebras for calculation of the detection type and detectability for TDFs

The TDF simulation for the given test pair $V = (v_1, v_2)$, will be carried out in the following steps. Both, the SAF simulation and TDF simulation are carried out by back-tracing, starting from outputs.

(1) First, using SAF simulator, a set of gates S will be determined for which the SAF fault is detected by the test pattern V_2.

(2) For each gate $G \in S$ we determine the type how the TDFs at the inputs $I \in G$ can be propagated to the gate output. This will be done by using algebra A1 in table "*Delay type calculation*" in Fig.5.

(3) For each input $I \in G$ of the gate $G \in S$ we calculate if the TDF is propagated to the output, and if it is we assign to this TDF its type calculated using algebra A2 in table "*Delay value calculation*" in Fig.5. If the calculated type of the TDF at the gate input will dominate over the type of the TDF at the output, the same type as the gate output has will be assigned to the gate input.

Example 1. Consider the procedure of using the algebras A1 and A2 for calculating the types of detected TDFs. For the output gate in Fig.3a, we have for the inputs: $x_4 = \varepsilon$, and $x_5 = h$, which gives by A1 the type = N (non-robust, yellow color). Then we find by A2 that there is no TDF tested for x_5, (the value ∅) but on the input x_4, the slow-to-rise TDF ($x_5 = \varepsilon$) will be tested non-robustly (the value N).

Example 2. Consider the circuit in Fig.6 to understand the whole procedure of TDF simulation.

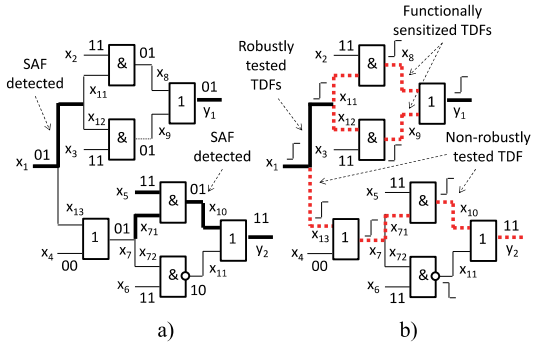


Fig.6. Example of conditional TDF simulation

In Fig.6a, the results of SAF simulation for the test vector V_2 are presented. The lines where SAF are detected are highlighted in bold. The SAF simulation is carried out by critical back-tracing for both test-patterns as explained in Section 2. In general, the SAF back-tracing is carried out in parallel, however, only the second vector of the test pair $V = (V_1, V_2)$ needs fault simulation. In Fig. 6b, the results of TDF analysis according to algebras A1 and A2 are presented. The analysis is carried out as well by back-tracing, starting always from the nodes were SAF is detected under V_2. The back-trace is continuing till the lines where no TDF is detected For example, the lines x_{11}, x_6, and x_{72} do not need TDF analysis.

5. Experimental Results

As experimental research we carried out TDF simulations for the ISCAS'85 circuits where we used test pairs derived from the test set generated for SAF faults. The results are presented in Table 1. From Table 1 we see that the TDF coverage calculation takes less time than the SAF simulation, and the more complex are circuits, the bigger is the difference. The speed of the SAF simulation is compared in Table 2 with different known fault simulators: FSIM [20], and two state-of-the-art commercial fault simulators C1 and C2 from major CAD vendors. Simulation times were calculated for the sets of random 10000 patterns. Experiments were run on a 1.5GHz Ultra SPARC IV+ workstation using SunOS 5.10.

Table 1. Experimental data of SAF and TDF simulation

Cir-cuit	SAF test			TDF test					
	Test length	Time ms	FC %	Test length	Time ms	Total	R	N	F
432	50	3	100	133	4.6	98.7	91.7	2.9	4.2
499	50	27	100	259	13.7	95.8	90.4	0.4	5.0
880	56	28	100	367	9.5	94.8	85.9	0.2	8.8
1355	100	48	100	320	21.0	95.7	91.3	0	4.4
1908	70	19	99.8	286	9.3	99.1	95.2	0	3.9
3540	178	167	100	680	29.3	100	99.6	0	0.4
5315	137	504	100	1159	72.5	100	99.6	0.1	0.3
6288	64	952	100	869	67.0	100	99.8	0	0.2

Table 2. Comparison of SAF simulation times

Circuts	Number of gates	SAF simulation time, s			
		Fsim	C1	C2	Proposed method
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

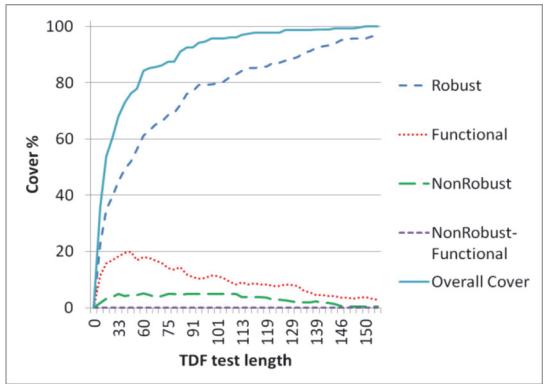


Fig.7. TDF covers as functions of the test length

We measured how the fault coverage was evolving as a function of the test length in the process of TDF simulation. The results for different TDF types for the circuit c432 are presented in Fig.7. In the simulation process we used the dominance relations between different fault sensitization types represented as: R > {N, F} > X, where N and F have the same rank. If the same TDF is detected under different sensitization conditions, then the dominating type will be assigned to the TDF. This explains why the curves of N and F type of TDFs in the beginning of the fault simulation process are rising and

later start to descend. In the end of the test, if not all TDFs can be tested robustly, they may be tested at-least in other N, F or X modes.

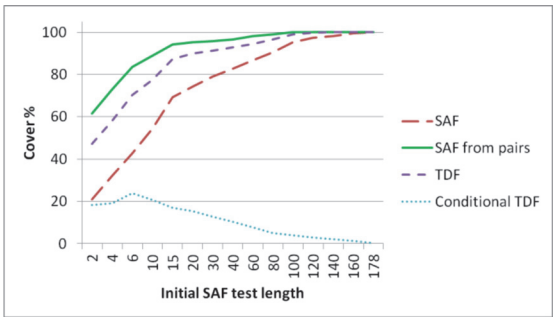


Fig.8. TDF covers as functions of the SAF cover basis

The Fig.8 demonstrates how the structure of the sensitization types for TDFs changes in dependence of the initial SAF test quality. We generated TDF test in a straightforward way from the initial SAF test. The lower was the SAF coverage of the initial SAF test set, the higher was the share of conditional TDF coverage compared to the robust test coverage (the group of conditional TDFs includes the TDFs that are detected only under N, F or X type of sensitization).

6 Conclusions

We proposed a new method for TDF simulation that extends the set of conditions under which TDFs can be detected. The following testing types were examined for TDF detection: robust, non-robust, functionally sensitized, and as a new type – non-robust functional sensitization. The target for test generation should be to get all TDFs robustly tested. However, if it is not achievable for all TDFs, a part of them still may be tested under alleviated conditions. The proposed simulation procedure consists of two phases: traditional SAF simulation and additional analysis under which conditions the TDFs may be detected. A 7-valued algebra is proposed for calculating separate TDF covers for the listed 4 sensitization conditions.

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] J.A. Waicukauski, E. Lindbloom, B.K. Rosen, V.S. Iyengar, "Transition fault simulation," *IEEE Design and Test*, pp. 32-38, April 1987.

[2] G.L. Smith, "Model for delay faults based upon paths," in *Proc. Int. Test Conf.*, 1985, pp.342-349.

[3] R.L. Wadsack et. al, "CMOS IC stuck-open fault electrical effects and design considerations," in *Proc. of IEEE ITC*, 1989, pp.423-430.

[4] I. Pomerantz, M. Reddy, "Transition path delay faults: a new path delay fault model for small and large delay defects," *IEEE Trans. On VLSI Systems,* Vol.16, No.1, 2008, pp. 98-107.

[5] I. Pomerantz, "Generation of mixed test sets for transition faults," IEEE Trans. On VLSI Systems, Vol.20, No.10, 2012, pp. 1895-1899.

[6] P. Gupta, M.S. Hsiao, "ALAPTF: A new transition fault model and the ATPG algorithm," in *Proc. of IEEE Int. Test Conf.,* 2004, pp.1053-1060.

[7] N. Jha, S.Gupta, *Testing of Digital Systems*. Cambridge, UK, Cambridge Univ. Press, 2003.

[8] I. Pomeranz, S.M. Reddy, "Hazard-Based Detection Conditions for Improved Transition Path Delay Fault Coverage," *IEEE Trans. On CAD of Integrated Circuits and Systems*, Vol.29, No. 9, 2010, pp. 1449-1453.

[9] I. Pomeranz and S. M. Reddy, "Hazard-based detection conditions for improved transition fault coverage of scan-based tests," IEEE Trans. Very Large Scale Syst., Vol. 18, No. 2, 2010, pp. 333–337.

[10] J.A.Waicukauski, et.al. Fault Simulation of Structured VLSI, VLSI Systems Design, Vol.6, No.12, pp.20-32, 1985.

[11] K.J.Antreich, M.H.Schulz, "Accelerated Fault Simulation and fault grading in combinational circuits", IEEE Trans. on CAD, Vol. 6, No. 5, pp.704-712, 1987.

[12] D.B. Armstrong. A deductive method for simulating faults in circuits. IEEE Trans. Comp., C-21(5), 464-471, 1972.

[13] E.G. Ulrich, T. Baker Concurrent simulator of nearly identical digital networks. IEEE Trans. on Comp., 7(4), pp.39-44, 1974.

[14] 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.

[15] S.Devadze, J.Raik, A.Jutman, R.Ubar. Fault Simulation with Parallel Critical Path Tracing for Combinational Circuits Using SSBDDs. LATW, 2006, pp.97-102.

[16] R.Ubar, S.Devadze, J.Raik, A Jutman. Parallel Fault Backtracing for Calculation of Fault Coverage. 13th ASPDAC, Seoul, Korea, 2008, pp. 667-672.

[17] 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.

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

[19] K.-T. Cheng, H.-C. Chen, "Classification and Identification of Nonrobust Untestable Path Delay Faults," *IEEE Trans. On CAD of IC and Systems*, Vol.15, No.8, 1996, pp. 845-853.

[20] 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, pp. 660-666.

Publication VI

R. Ubar, **J. Kõusaar**, S. Devadze, J. Raik. Transition Delay Fault Simulation with Parallel Critical Path Back-Tracing and 7-valued Algebra. Microprocessors and Microsystems - MICPRO, Elsevier, Volume 39, Issue 8, Nov. 2015, pp. 1130-1138.

Transition Delay Fault Simulation with Parallel Critical Path Back-Tracing and 7-valued Algebra

Jaak Kõusaar, Raimund Ubar, Sergei Devadze, Jaan Raik
Department of Computer Engineering
Tallinn University of Technology, Tallinn, Estonia
jaak.kousaar@gmail.com, raiub@pld.ttu.ee

***Abstract*—A new method is presented for simulating of Transition Delay Faults (TDF) based on the parallel exact critical path tracing for Stuck-at Fault (SAF) analysis and subsequent TDF reasoning. A method is proposed to extend the TDF model, traditionally considered as a class of robustly tested delay faults, to a class of TDFs with extended detection conditions. Three known fault classes of delay fault sensitization are considered: robust, non-robust and functional sensitization of delay faults. Additionally, a new fourth fault class is introduced, called non-robust functionally sensitized delay fault. A novel fault analysis algorithm based on 7-valued algebra is presented, which delivers the fault coverage for all mentioned four types of TDFs.**

Keywords – transition delay faults, non-robust and functional sensitization, critical path fault tracing, 7-valued algebra

1 INTRODUCTION

The purpose of delay testing is to ascertain that manufactured digital circuits meet their timing specifications. Delay faults can be modeled in different ways, among which the most common models are transition delay fault (TDF) model [1], and path delay fault (PDF) model [2]. The TDF model captures large delay defects that affect single locations in the circuit whereas the PDF model captures small delays, such that each one by itself may not cause the circuit to fail, but their cumulative effect along a path from inputs to outputs may result in faulty behavior. The TDF is as well used as a logic model for stuck-open faults in CMOS circuits, which either suppress or delay the occurrence of certain transitions [3].

The main advantage of the TDF model is the linearity of the number of faults in terms of the number of connections in gate-level circuits. Another advantage is that the stuck-at-fault (SAF) test generation and fault simulation tools can be directly used for handling TDFs. The disadvantage of TDFs is that the expectation that the delay fault in a single location is large enough for propagating up to the observation points might not be realistic. On the other hand, the disadvantages of the PDF model is that the number of paths in circuits is huge, the number of robustly testable paths is in general very small, and as shown in [4], the non-robust tests may not detect delays in certain situations where the cumulative effect of small delays would be sufficient to be tested non-robustly.

In [4,5] a novel *Transition Path Delay Fault* (TPDF) model was proposed where the ideas of TDF and PDF models are combined to improve the capability of TDFs to detect the delays by combining tests for TDFs along the target paths.

In [6] a new TDF model is proposed called *As Late As Possible Transition Fault* (ALAPTF). The model aims at detecting cumulatively smaller delays, which will be missed by both the traditional TDF and the PDF models. The model makes sure that each transition is launched as late as possible at the TDF site, accumulating the small delay defects along its way.

Both approaches [4] and [6] are based on the idea of creating paths along which the TDFs are consecutively tested.

In both approaches we need to have continuous activated paths or segments of paths along which the TDF will robustly propagate. In general, such paths may be missing, or it may be difficult to create paths along which all the TDFs are detected. In these cases, it may be reasonable to alleviate the constraints on consecutive propagation of TDFs robustly along the tested path, so that in some gates or segments the TDFs may propagate non-robustly or along discontinuous paths.

In [4], it was shown that a test for TPDF corresponds to a type of strong non-robust test [7] for a conventional PDF associated with the same path, however, with the difference that the TPDF test additionally detects the TDFs on every line of the path.

On the other hand, a strong non-robust test satisfies the weak non-robust propagation conditions [7], but not the opposite. As shown in [8], this results in a large gap in the number of detectable faults between TPDFs as defined in [4] and conventional PDFs when the weak non-robust propagation conditions are used for them. To bridge this gap, in [8], the detection conditions for TPDFs used in [4] were extended with hazard-based detection conditions for transition faults defined in [9]. When the hazard-based detection conditions are allowed for the transition faults included in a TPDF, it is not necessary to create transitions on every line of the path, rather it is also possible for lines on the path to have hazards. When some of the TDFs associated with a TPDF are detected under the hazard-based detection conditions, the resulting test will be a type of weak non-robust test.

In this paper, we further alleviate the conditions of detecting TDFs in addition to the hazard-based conditions (let us call them non-robust detection conditions for TDFs). We introduce into the detection conditions for TDFs so called

functional sensitization [7] to handle multiple TDFs and a new type of conditions that we call *non-robust functional sensitization* conditions which allows multiple TDF detection in hazard-based conditions. To determine the detectability of TDFs under mentioned conditions, we introduce 7-valued algebra which will support SAF simulation, and allows to classify all the detected TDFs into the classes of robustly detected, non-robustly detected, detected under functional sensitization conditions or under non-robust functional sensitization conditions.

We carry out the TDF simulation in two phases: traditional SAF simulation and additional check if the conditions needed for TDF detection are satisfied.

SAF simulation can be carried out in different ways: by parallel pattern single fault propagation [10], critical path tracing [11], deductive [12], or concurrent [13] fault simulation. The last three approaches are based on logic reasoning rather than simulation, and hence, are very powerful since all the detectable faults are calculated by a single run of the current test pattern. What they cannot do is to produce reasoning for many test patterns in parallel.

The Critical Path Tracing (CPT) approach consists of simulating the fault-free circuit and using the computed signal values for back-tracing all sensitized paths from primary outputs to primary inputs, to determine all the faults detectable by the given test pattern. The back-trace continues until the paths become non-sensitive or end at network primary inputs. Faults on the sensitive (critical) paths are detectable by the test. In the conventional CPT approach the detectable faults are found by a single run, however the precise results are only guaranteed in fan-out free circuits.

A modified critical path tracing technique that is linear time, exact, and complete is proposed in [14]. However, the rule based strategy used in this approach does not allow simultaneous parallel analysis of many patterns. Parallel critical path backward tracing in fan-out free regions combined with parallel simulation of stem faults was investigated in [15]. In [16], the concept of exact parallel critical path tracing was generalized for SAF cover analysis beyond the fan-out free regions, and in [17], the method was extended for simulation of a large fault class called X –fault.

In this paper we adjust the method proposed in [17] for parallel critical path tracing of TDFs, and we extend the detection conditions for TDFs compared to that proposed in [8,9]. Both phases of TDF simulation will be carried out by critical path backward tracing, the phase of SAF analysis is carried out in parallel for many patterns simultaneously, and the TDF type analysis is carried out pattern by pattern using the 7-valued algebra developed in the paper.

The organization of the paper is as follows. First, in Section 2, the concept of fast parallel critical path tracing method for calculating the detection of SAFs is described. In section 3, an extension of the class of TDF faults is introduced and explained. Section 4 presents a novel 7-valued algebra for determining the detectability of different types of TDFs, and in Section 5 a general procedure for novel TDF simulation is described. Section 6 presents experimental results, and Section 5 concludes the paper.

2 SAF SIMULATION WITH CRITICAL PATH TRACING

In the first phase of TDF simulation we determine the detected SAF faults in the circuit for the given set of test patterns. For this purpose we use the idea of parallel backward critical path tracing used in [16, 17]. In the following, the method of SAF simulation based on this idea is shortly described.

Consider a network of Fan-out-Free Regions (FFR) where each FFR is represented as a Boolean function.

$$y = F(x_1, \ldots, x_i, x_j, \ldots x_n) = F(X),$$

and $X = x_1, x_2, \ldots x_n$ is the input vector of the FFR. Such a network of five 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.

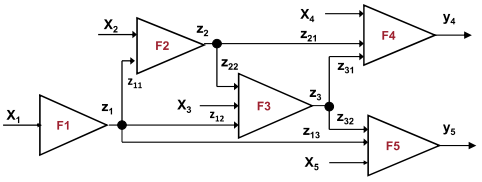


Fig.1. *Combinational circuit with five FFRs*

The task of SAF simulation can be referred to calculation of Boolean derivatives: if $\partial y/\partial x_j = 1$ then the fault is propagated from x_j to y. The calculation of derivatives can be performed in parallel for a subset of test patterns, so that each bit in a computer word will correspond to one test pattern. In order to extend the parallel critical path tracing beyond the fan-out free regions we use the concept of partial Boolean differentials [18].

Consider an FFR F, where a group of paths is re-converging from the same node x, as shown in Fig.2. Denote the Boolean function that represents the gate F as $y = F(x_1, \ldots, x_i, x_j, \ldots x_n)$.

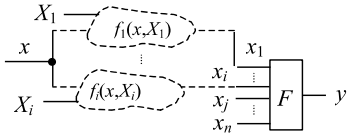


Fig.2. *Reconvergent FFR in a circuit*

Assume that the input variables $x_1,\ldots,x_i$ of the FFR F are connected to the fan-out stem x via other FFRs represented by Boolean functions

$$x_1 = f_1(x,X_1),\ \ldots,\ x_i = f_i(x,X_i),$$

where the variables of vectors X_i correspond to the nodes in the circuit which do not have paths to x. Then all possible fault propagation conditions from x to y can be described by full Boolean differential [18]:

$$dy = dF = y \oplus F((x_1 \oplus dx_1),\ldots,(x_j \oplus dx_i),\ (x_j \oplus dx_j),\ldots,(x_n \oplus dx_n)) \qquad (1)$$

where the Boolean variables dx and dy denote the erroneous changes of the values of x and y, respectively, caused by a propagated fault. By dx we denote the change of the value of x because of the influence of a fault at x, and $dy = 1$ if some erroneous change of the values of arguments of the function (1) causes the change of the value of y, otherwise $dy = 0$.

In [16] we have shown that if a SAF at y is propagated and detected by a test pattern at primary outputs then the fault at the fan-out stem x is as well detected if

$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}),\ldots,(x_i \oplus \frac{\partial x_i}{\partial x}),x_j,\ldots,x_n) = 1 \cdot \qquad (2)$$

The formula (2) taken in the vector form can be simplified as

$$\frac{\partial y}{\partial x} = y \oplus F(X' \oplus \frac{\partial X'}{\partial x}, X'') \qquad (3)$$

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, for calculating if the fault on z_2 can be detected on y_4 in Fig.1, we can check if

$$\frac{\partial y_4}{\partial z_2} = y \oplus F(X_4, z_{21} \oplus 1, z_{31} \oplus \frac{\partial z_3}{\partial z_2}) = y \oplus F(X_4, \overline{z_{21}}, z_{31} \oplus \frac{\partial z_3}{\partial z_2}) = 1$$

From the differential equation (2), a method results which generalizes the parallel exact critical path tracing beyond FFRs. As all the operations in (2) are Boolean, they can be carried out in parallel.

In a general case of nested re-convergences, the formula (2) can be used recursively [16,17]. If a SAF is detected by a test pattern on the output y of a sub-circuit in Fig.3 with two nested re-convergences, $y = F_y(x_1,z,X_y)$ and $z = F_z(x,X_z)$, where X_y and X_z are not depending on x, then the fault at the common re-convergent fan-out stem is also detected if

$$d_x y = y \oplus F_y(x_1 \oplus \frac{\partial x_1}{\partial x}, z \oplus d_x F_z, X_y) = 1 \qquad (4)$$

.

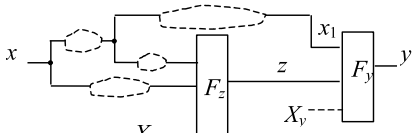


Fig.3. *Nested re-convergences in a circuit*

The formula (4) can be used for calculating the influence of the fault (or erroneous signal) at the common fan-out stem x on the output y of the reconvergent fan-out region by consecutive calculating of Boolean derivatives over related FFR chains starting from x up to y. For that purpose, for each fan-out stem involved in a reconvergent subnetwork, the corresponding formulas like (4) should be constructed. 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.

The whole process of fault simulation can be carried out in two phases. In the first phase, parallel logic simulation is carried out to determine the logic values for all nodes in the circuit which are needed to carry out the fault simulation In the second phase the fault simulation using the formulas presented above is carried out for the simulated bunch of test patterns.

Both phases can be carried out on the levelized topological model. During the first phase, the logic simulation is carried out level by level in direction from inputs to outputs whereas during the second phase the fault simulation is carried out in the opposite backward direction from outputs to inputs. The levelized model for the circuit given in Fig.1 is represented in Fig.4.

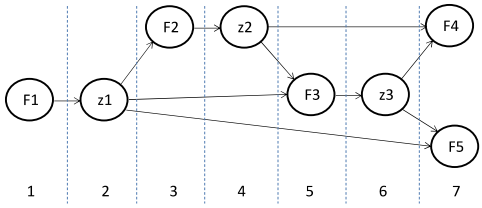


Fig.4. *Levelized topological model of the circuit in Fig.1 used for logic and fault simulation phases*

For this example, in the first phase of logic simulation, the levels denoted by F are representing the logic signal calculations, and the levels denoted by z are representing the operations for updating the values of variables in fan-out nodes of the circuit. In the second phase of fault simulation, the levels denoted by F are representing the tasks of finding the detected faults inside the FFRs whereas the levels denoted by z are representing the operations for finding at which patterns the faults are detected in fan-out nodes of the circuit.

Example 1. To demonstrate the second phase of the fault simulation process, introduce the following notations for the formulas above which are used for calculating the Boolean derivatives:

- (x, y) – for $\partial y/\partial x$,
- $R_{xy}((x, x_1), \ldots (x, x_k))$ – for the general case (3), where $X' = (x_1,\ldots,x_k)$,
- Dx – vector which shows if the fault at the node x is detected or not detected at any circuit output,

An example of a computational model of fault simulation for the circuit in Fig.1 is presented in Table 1.

***Table 1**. Example of critical path 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 [16]. The algorithm has linear complexity. However, the complexity of the computational model and the related fault simulation speed depends essentially on the structure of the circuit. However, as shown in the papers [16,17], the speed of the fault simulation for different fault models (like conditional SAFs and X-faults) by the proposed parallel critical path tracing method outperforms the speed of the fault simulators of major CAD vendors.

As the result of the described parallel critical path tracing procedure, the following two tables are created:

(1) Fault table $FT = \| r_{i,j} \|$, $r_{i,j} \in \{0,1,\varnothing\}$ where i and j denote the numbers of simulated test patterns and nodes in the circuit, respectively; $r_{i,j} = 0$ (1) if the fault stuck-at 0(1) on the node j is detected by the pattern i, and $r_{i,j} = \varnothing$ if none of these faults is detected on the node j by the pattern i.

(2) Test pattern table $TP = \| p_{i,j} \|$ where $p_{i,j} \in \{0,1\}$ is the signal value on the node j produced by the pattern i.

The data of tables FT and TP will be used for mapping SAF faults from FT to TDF candidates to be confirmed during the second phase of TDF simulation.

3. Extension of the Class of Transition Delay Faults

To detect a TDF, we have to apply two input patterns, $V = (V_1, V_2)$, so that V_1 will initialize the circuit, while V_2 will activate the fault and propagate its effect to some primary output. In the first phase of SAF simulation, we determine all the nodes in the circuit which were tested for SAF. For example, if the node x_i was tested for SAF $x_i \equiv 1$, then this node is the candidate for detecting the TDF "slow-to-fall" transition at this node, and if the node x_i was tested for SAF $x_i \equiv 0$, then this node is the candidate for detecting the TDF "slow-to-rise" transition at this node.

In conventional TDF testing it is needed that a test pair $V = (V_1, V_2)$ will assign to the node x_i different values, to initiate a transition, either $0 \rightarrow 1$ for detecting the TDF "slow-to rise", or $1 \rightarrow 0$ for detecting the TDF "slow-to-fall". These conditions we classify as conditions to detect the TDF robustly. To determine, if the TDF fault $v \rightarrow v$' is detected on the line x_i, where $v \in \{0,1\}$, and v' means inverted v, it is needed that the test pattern V_1 will assign $x_i = v$, the test pattern V_2 will assign $x_i = v$', and the SAF fault $x_i \equiv v$ is detected by V_2 at any primary output of the circuit.

In general case, e.g. when using the hazard-based detection conditions for the TDF $v \rightarrow v$' on the node x_i, it is allowed to have a hazard or a pulse on the line x_i, under which x_i makes the transitions $v' \rightarrow v \rightarrow v'$ during the second pattern V_2 of the test. The fault is assumed to delay the $v \rightarrow v$' transition of the pulse. Thus, in the presence of fault, x_i is assumed to make the transition $v' \rightarrow v$, and stay at v. So, the fault can be detected by a test that creates the pulse on x_i and propagates the pulse under the second test pattern to an output. In other words, at the hazard-based conditions it is not necessary to create a transition on every line of the path to detect a TDF. It is also possible for lines on the path to have hazards. The TPDFs detected under hazard-based conditions correspond to a type of weak-non-robust test. We call this case as a non-robustly tested TDF.

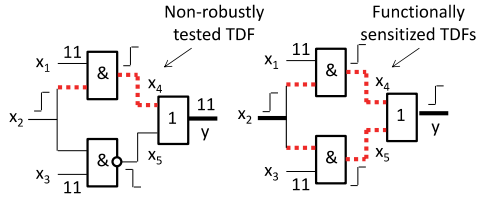


Fig.5. TDF is propagated along discontinuous paths

Example 2. Consider an example in Fig.5 where a test pair $V = (V_1, V_2)$ is applied on the inputs with transition $0 \rightarrow 1$ on x_2. Assume that on both of the circuits the SAF faults at the outputs y are detected by V_2. In both circuits, the transition on the node x_2 is propagating further through two paths which re-converge at the node y.

In the case of Fig.5a, the fault effect on x_2 disappears, and does not reach the output y, because of different inversion parities on the re-converging paths. This is the reason why SAF $x_2 \equiv 0$ is not detected under V_2. Consequently, the TDF $0 \rightarrow 1$ on x_2 is not propagating to y, and the TPDF on the path x_2 - x_4 - y is not detected as well. However, the TDF $0 \rightarrow 1$ on x_4 can propagate non-robustly to y, and hence detected non-robustly. As the result, the TPDF on the path x_2 - x_4 - y can be detected non-robustly as well.

In Fig.5b, the TDF 0 → 1 on x_2 propagates to y, and is detected. However, the TDFs on x_4 and x_5 are not detected, because the SAFs $x_4 \equiv 0$ and $x_5 \equiv 0$ are not propagated under V_2. This means that there is no path in the circuit, where the TDFs are detected on all lines of the path, and consequently no TPDFs are detected in the circuit under test pair V. On the other hand, the TDFs at the nodes x_4 and x_5 are sensitized functionally [7], and can propagate to y as a multiple delay fault. This means that on both paths from x_2 to y, the TDFs are detected on all lines under alleviated conditions (under functional sensitization conditions on the lines x_4 and x_5), and hence, the TPDFs for the paths from x_2 to y are as well detected under functional sensitization conditions. □

To define the testing possibilities for TDFs, and alleviate in the same time the constraints on detection of TDFs, we introduce the following types of TDFs: *robustly tested* TDF, i.e. the conventional case (denote it as R-type), *non-robustly tested* TDF (N-type), *functionally sensitized* TDFs [7], i.e. the multiple delay fault case (F-type), and a new type of *non-robustly functionally sensitized* TDFs (X-type).

These types of testing of TDFs are illustrated in Fig.6. For all the cases depicted, it is assumed that the SAF fault on the output of the gate is detected under the test vector V_2.

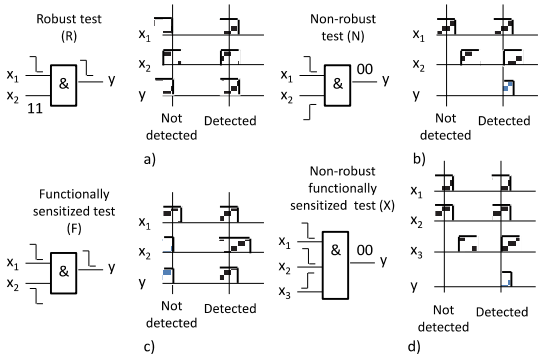


Fig.6. *Four types of conditional detecting of TDFs*

In all the cases in Fig.6 the TDF on the input x_1 is under test. The first diagram shows the waveforms of signals for the case where the delay fault on x_1 is missing or is not detected, and the second diagram shows the case where the delay fault is detected. In Fig.6a, the TDF on x_1 is tested robustly (conventional case, or TDF of type R). Fig.6b shows the case of non-robust testing (the TDF of type N). The TDF on x_1 may not be detected if the transition 0 → 1 on x_2 is late. If this transition will be not delayed, or is delayed less than the transition on x_1, the TDF will be detected under the hazard-based conditions [12]. In Fig.6c, the TDF on x_1 is tested under functional sensitization conditions (the TDF of type F) [11]. This is the case when only multiple TDFs can be detected, i.e. the delay of the transition 1→ 0 on x_1 can be detected only if the same transition on x_2 is as well delayed. In the conventional meaning of TDF testing, the TDFs on the lines x_1 and x_2 can be regarded as untestable faults. Finally, in Fig.6d, the combination of the cases in Fig.6b and Fig.6c is considered (the TDF of type X). This is a new type of TDFs introduced in the paper. The multiple TDFs on x_1 and x_2 may not be detected if the transition 0 → 1 on x_3 is late. If this transition will not be delayed, or is delayed less than the both transitions on x_1 and x_2, the TDF will be detected.

The TPDF model [4] for a path P requires that all the TDFs on the path P were detected. The detection of the TDF types like N, F and X allows alleviating the conditions under which the TDF will be considered as detected. However, the TDF detection conditions defined by N, F and X types are not guaranteed to happen. The goal of alleviation of the TDF detection conditions is to bridge the gap between the numbers of detectable TPDFs [4], and conventional PDFs, in a similar way as it was proposed in [8] by introducing the hazard-based TDF detection conditions (the type N of TDFs). In this paper we introduced additionally two more sensitizing conditions (F- and X-types of TDFs) to improve the exactness of TPDF coverage.

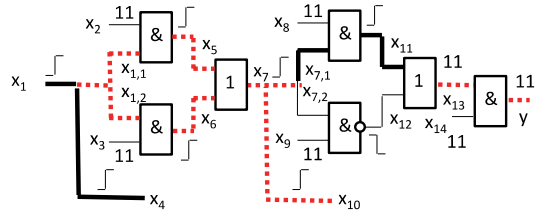


Fig.7. *Detection of different types of TDFs*

Table 2. *Detected TDF faults*

x_i	x_1	$x_{1,1}$	$x_{1,2}$	x_4	x_5	x_6	x_7	$x_{7,1}$	x_{10}	x_{11}	x_{13}	y
1	**R**	F	F	**R**	F	F	**R**	**R**	**R**	**R**	∅	∅
2	**R**	X	X	**R**	X	X	N	**R**	N	**R**	∅	∅

Example 3. Consider circuit in Fig.7 where all the signal values of the given test pair are depicted. In Table 2, the detected TDF faults are shown for two cases: 1) the line x_{10} is tested via a sub-circuit, not shown in the Figure for the R-type of TDF (TDF/R), similarly as in the traditional case of TDFs, and 2) x_{10} is tested for TDF/N. Only the lines x_1, x_4, $x_{7,1}$, and x_{11} are tested in both cases robustly for TDF/R, as in the conventional case. All other lines are not tested for TDFs in the traditional meaning of TDFs. However, if considering the 3 types of dynamic faults (types N, F, and X) introduced in the previous section, then we see that a lot of other lines can be as well tested for TDFs at different sensitizing conditions. If line x_7 is tested robustly (case 1), then the lines $x_{1,1}$, $x_{1,2}$, x_5, and x_6 are tested for TDF/F, otherwise for TDF/X (case 2). Explanation of that will be given in the next section.

4 7-VALUED ALGEBRA FOR TDF SIMULATION

For simulating of TDFs and qualifying the types of the TDFs in accordance with the conditions they can be detected, we have developed two 7-valued algebras A1 and A2 as shown

in Fig.8 and Fig.9 for the gates OR/NOR, and AND/NAND, respectively. The algebras are developed for two-input gates. If the number of inputs of the gate is bigger than two, then we have to represent the multi-input gate by an equivalent sequence of two-input gates. The algebras are defined for the set of values (symbols) {0,1,ε, *h*, *N*, *F*, *X*}, with interpretations explained in Fig.10, and by the binary calculation operators defined by Tables A1 and A2 in Fig.8, and Fig.9.

A1: Next signal propagation type

OR	Current signal propagation type						
Transition type	0	1	ε	h	N	F	X
0	0	1	ε	h	N	F	X
1	1	1	1	1	1	1	1
ε	ε	1	F	N	X	F	X
h	h	1	N	h	N	X	X

A2: Delay fault type

OR	Tran-sition type	
Signal propagation type	ε	h
0	∅	∅
1	∅	∅
ε	R	∅
h	∅	R
N	N	∅
F	F	∅
X	X	∅

Fig.8. *Two 7-valued algebras for the gates OR/NOR*

A1: Next signal propagation type

AND	Current signal propagation type						
Transition type	0	1	ε	h	N	F	X
0	0	0	0	0	0	0	0
1	0	1	ε	h	N	F	X
ε	0	ε	ε	N	N	X	X
h	0	h	N	F	X	F	X

A2: Delay fault type

AND	Tran-sition type	
Signal propagation type	ε	h
0	∅	∅
1	∅	∅
ε	R	∅
h	∅	R
N	∅	N
F	∅	F
X	∅	X

Fig.9. *Two 7-valued algebras for the gates AND/NAND*

Interpretation of symbolic values			
Transition types		Delay fault types	
0	00/No transition	F	Functionally senzitized delay
1	11/No transition	N	Non-robust testable delay
h	10 /Falling transition	X	Non-robust functionallly sensitized delay
ε	01 / Rising transition	R	Robust testable delay

Fig.10. Interpretation of symbolic values of algebras A1 and A2

The symbols 0, 1, ε, and *h* denote the signals and transitions on the inputs of the gate under analysis assigned by the test vector $V = (V_1, V_2)$. The symbols 0, 1 represent the steady signals 00 and 11, respectively, and the symbols ε and *h* represent the transitions 01 and 10, respectively. The symbols *R*, *N*, *F* and *X* denote the types of the conditions under which the TDFs can be detected. The symbol ∅ means that the TDF on the gate input under analysis is not tested. The symbol R is equivalent to the traditional case of testing TDFs.

We put the fault types in the following dominance relation R > N, R > F, R > X, N > X, F > X. The higher is the ranking in this relation, the looser are the sensitization conditions for TDF detecting. Hence, it would be always advisable to choose the type of highest ranking if there is a possibility for a choice.

We have introduced a sequential algebra for analysis of the input signals of the gate because in the case of multi-input gates the signals impact is not independent of other signals when we are calculating the type of the TDF propagation conditions. In Fig.11, it is shown how the multiple input gate is represented as a sequence of 2-input gates for sequentially using Algebra A1 for calculation of the signals propagation type for the original multi-input gate. Then, by Algebra A2, the detectable TDF types for the inputs of the gate are calculated.

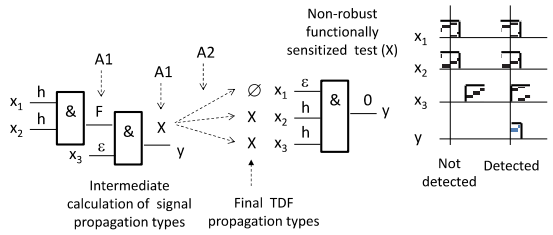


Fig.11. *Using sequentially the Algebras A1 and A2 for TDF type calculation*

The TDF simulation for the given test pair $V = (v_1, v_2)$, will be carried out according to the Procedure 1 in the following. Both, the SAF simulation and TDF simulation are carried out by back-tracing, starting from outputs towards inputs. We assume that all the gates in the circuit are ranked in such a way that the outputs will have the highest rank, and the next rank can be assigned to a gate input line only if the gate already has assigned a rank.

5 General Procedure of TDF Simulation

In the following we present a general procedure for determining the values of different types of TDF faults using the Algebras A1 and A2.

Procedure 1.

(1) First, using SAF simulator, a set of lines *L* will be determined for which the SAF fault is detected by the test pattern V_2.

(2) For all the lines $l \in L$ where $V_1(l) \neq V_1(l)$, the TDF of type R will be fixed as detected (this determines the traditional TDF coverage).

(3) For each gate with output line $l \in L$, taken in the order of line ranking, we determine the types of propagation of TDFs from the inputs of the gate to its output *l*. If an input line, not detected for TDF/R, will be assigned with a type TDF/D where

D ∈{N, F, X}, we include this line into L. This step will be carried out by iterative using of Algebras A1 and A2.

(4) In the process of simulating of the given test sequence we update the current fault coverage by the simulation results for the current test pair in the following way:

- for the lines where the TDF was detected the first time, we mark the line as detected for the given TDF type;

- if a line was already detected earlier for any TDF type, we choose the type which has higher ranking in the dominance relation: R > (F, N) > X.

When analyzing a gate in Step 3 of Procedure 1 we will take into account the following aspects.

(1) If the input of the gate is already fixed as detected for TDF/R, then we don't need to recalculate the TDF type.

(2) If for the output line $l \in L$ of the gate a type D ∈{N, F, X} is assigned, then we apply the transformation of the sub-circuit in opposite direction, compared to the case illustrated in Fig.11, and substitute the sub-circuit by an equivalent multi-input gate. The problem is illustrated in Fig.12. Let the output line of the NAND gate has the type N. First, we substitute the whole sub-circuit by the equivalent OR gate with inverted inputs x_1 and x_2. Then we interpret the value N as the result of the analysis of the OR gate after using Algebra A1 for the inputs x_2 and x_3. Using now iteratively Algebra A1 for the input x_1, we get the type X for the equivalent gate. Finally, using now Algebra A2, we calculate the final types for x_1 and x_2. Here we see that the type N for the NAND gate has changed now to the weaker type X at the inputs. This is the same effect that we noticed in the Example 3: the lines $x_{1,1}$, $x_{1,2}$, x_5, and x_6 in Fig.7 got the weaker types (with lower ranking) than the line x_7 (see also in Table 2).

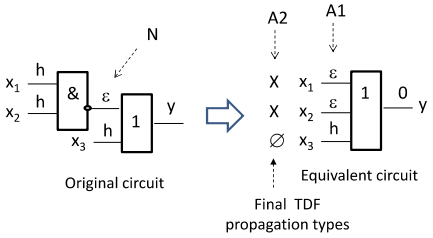


Fig. 12. *Backward propagation of TDF types*

(3) For the lines with fan-out we assign the strongest type over the branches of the stem.

Example 4. Consider the procedure of using the algebras A1 and A2 for calculating the types of detected TDFs. For the output gate in Fig.13a, we have for the inputs: $x_4 = \varepsilon$, and $x_5 = h$, which gives by A1 the type = N (non-robust, yellow color). Then we find by A2 that there is no TDF tested for x_5, (the value ∅) but on the input x_4, the slow-to-rise TDF ($x_5 = \varepsilon$) will be tested non-robustly (the value N).

Example 2. Consider now the circuit in Fig.13 to understand the whole procedure of TDF simulation described as Procedure 1.

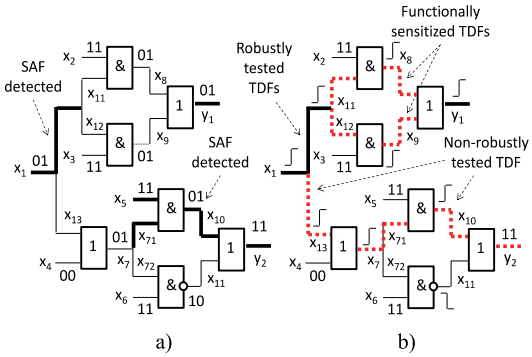


Fig.13. *Example of conditional TDF simulation*

In Fig.13a, the results of SAF simulation for the test vector V_2 are presented. The lines where SAF are detected are highlighted in bold. The SAF simulation is carried out by critical back-tracing for both test-patterns as explained in Section 2. In general, the SAF back-tracing is carried out in parallel, however, only the second vector of the test pair $V = (V_1, V_2)$ needs fault simulation. In Fig. 13b, the results of TDF analysis according to algebras A1 and A2 are presented. The analysis is carried out as well by back-tracing, starting always from the nodes were SAF is detected under V_2. The back-trace is continuing till the lines where no TDF is detected For example, the lines x_{11}, x_6, and x_{72} do not need TDF analysis.

6. Experimental Results

As experimental research we carried out TDF simulations for the ISCAS'85 circuits where we used test pairs derived from the test set generated for SAF faults. The results are presented in Table 3. From Table 3 we see that the TDF coverage calculation takes less time than the SAF simulation, and the more complex are circuits, the bigger is the difference. The speed of the SAF simulation is compared in Table 4 with different known fault simulators: FSIM [20], and two state-of-the-art commercial fault simulators C1 and C2 from major CAD vendors. Simulation times were calculated for the sets of random 10000 patterns. Experiments were run on a 1.5GHz Ultra SPARC IV+ workstation using SunOS 5.10.

Table 3. *Experimental data of SAF and TDF simulation*

Cir-cuit	SAF test			TDF test					
	Test length	Time ms	FC %	Test length	Time ms	Total	R	N	F
432	50	3	100	133	4.6	98.7	91.7	2.9	4.2
499	50	27	100	259	13.7	95.8	90.4	0.4	5.0
880	56	28	100	367	9.5	94.8	85.9	0.2	8.8
1355	100	48	100	320	21.0	95.7	91.3	0	4.4
1908	70	19	99.8	286	9.3	99.1	95.2	0	3.9
3540	178	167	100	680	29.3	100	99.6	0	0.4
5315	137	504	100	1159	72.5	100	99.6	0.1	0.3
6288	64	952	100	869	67.0	100	99.8	0	0.2

Table 4. *Comparison of SAF simulation times*

Circuts	Number of gates	SAF simulation time, s			
		Fsim	C1	C2	Proposed method
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

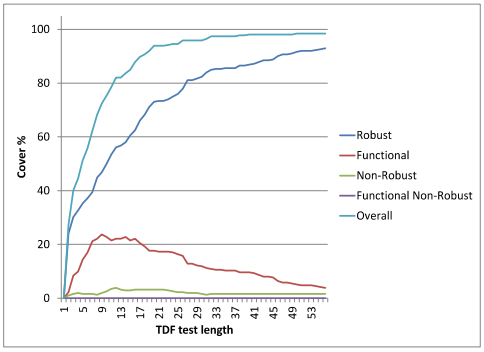


Fig.14. *TDF covers as functions of the test length*

We measured how the fault coverage was evolving as a function of the test length in the process of TDF simulation. The results for different TDF types for the circuit c432 are presented in Fig.14. In the simulation process we used the dominance relations between different fault sensitization types represented as: R > {N, F} > X, where N and F have the same rank. If the same TDF is detected under different sensitization conditions, then the dominating type will be assigned to the TDF. This explains why the curves of N and F type of TDFs in the beginning of the fault simulation process are rising and later start to descend. In the end of the test, if not all TDFs can be tested robustly, they may be tested at-least in other N, F or X modes.

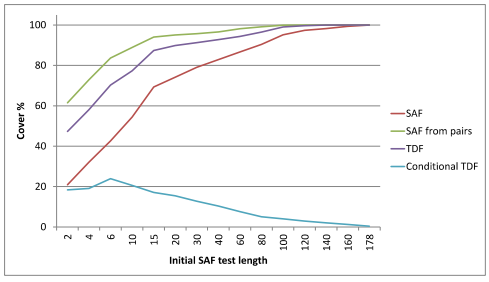


Fig.15. *TDF covers as functions of the SAF cover basis*

The Fig.15 demonstrates how the structure of the sensitization types for TDFs changes in dependence of the initial SAF test quality. We generated TDF test in a straightforward way from the initial SAF test. The lower was the SAF coverage of the initial SAF test set, the higher was the share of conditional TDF coverage compared to the robust test coverage (the group of conditional TDFs includes the TDFs that are detected only under N, F or X type of sensitization).

7 Conclusions

We proposed a new method for TDF simulation that extends the set of conditions under which TDFs can be detected compared to the state-of-the-art where only the hazard-based detection conditions have been considered [9].

The following new testing types were investigated for TDF detection, besides robust (traditional TDF model) and non-robust sensitization approaches: functionally sensitized, and as a new advanced propagation type – non-robust functional sensitization. Taking into account the two new sensitization conditions (F- and X-types) of TDFs allows to improve the accuracy of TPDF coverage, and to contribute to the advanced TPDF model developed in [4,9].

The target for test generation should be to get all TDFs robustly tested. However, if it is not achievable for all TDFs, a part of them still may be tested under alleviated conditions. The proposed simulation procedure consists of two phases: traditional SAF simulation and additional analysis under which conditions the TDFs may be detected.

For robustly detectable TDF simulation, a fast pattern parallel exact critical path tracing method was developed, which surpasses in performance 2-4 times the state-of-the-art SAF simulators.

For simulating other types of conditionally detectable TDFs, introduced in the paper, a sequential 7-valued algebra was proposed. The fault simulation is as well based on the backtracing principle, which allows to determine all the detected TDFs under different propagation conditions by a single run of the given test pair. The proposed method has a linear complexity.

Acknowledgements: The work was supported by EU FP7 STREP project BASTION, Estonian ICT project FUSETEST, by EU through the European Structural 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, E. Lindbloom, B.K. Rosen, V.S. Iyengar, "Transition fault simulation," *IEEE Design and Test*, pp. 32-38, April 1987.

[2] G.L. Smith, "Model for delay faults based upon paths," in *Proc. Int. Test Conf.*, 1985, pp.342-349.

[3] R.L. Wadsack et. al, "CMOS IC stuck-open fault electrical effects and design considerations," in *Proc. of IEEE ITC*, 1989, pp.423-430.

[4] I. Pomerantz, M. Reddy, “Transition path delay faults: a new path delay fault model for small and large delay defects,” *IEEE Trans. On VLSI Systems,* Vol.16, No.1, 2008, pp. 98-107.

[5] I. Pomerantz, “Generation of mixed test sets for transition faults,” IEEE Trans. On VLSI Systems, Vol.20, No.10, 2012, pp. 1895-1899.

[6] P. Gupta, M.S. Hsiao, “ALAPTF: A new transition fault model and the ATPG algorithm,” in *Proc. of IEEE Int. Test Conf.,* 2004, pp.1053-1060.

[7] N. Jha, S.Gupta, *Testing of Digital Systems*. Cambridge, UK, Cambridge Univ. Press, 2003.

[8] I. Pomeranz, S.M. Reddy, “Hazard-Based Detection Conditions for Improved Transition Path Delay Fault Coverage,” *IEEE Trans. On CAD of Integrated Circuits and Systems*, Vol.29, No. 9, 2010, pp. 1449-1453.

[9] I. Pomeranz and S. M. Reddy, “Hazard-based detection conditions for improved transition fault coverage of scan-based tests,” IEEE Trans. Very Large Scale Syst., Vol. 18, No. 2, 2010, pp. 333–337.

[10] J.A.Waicukauski, et.al. Fault Simulation of Structured VLSI, VLSI Systems Design, Vol.6, No.12, pp.20-32, 1985.

[11] K.J.Antreich, M.H.Schulz, "Accelerated Fault Simulation and fault grading in combinational circuits", IEEE Trans. on CAD, Vol. 6, No. 5, pp.704-712, 1987.

[12] D.B. Armstrong. A deductive method for simulating faults in circuits. IEEE Trans. Comp., C-21(5), 464-471, 1972.

[13] E.G. Ulrich, T. Baker Concurrent simulator of nearly identical digital networks. IEEE Trans. on Comp., 7(4), pp.39-44, 1974.

[14] 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.

[15] S.Devadze, J.Raik, A.Jutman, R.Ubar. Fault Simulation with Parallel Critical Path Tracing for Combinational Circuits Using SSBDDs. LATW, 2006, pp.97-102.

[16] R.Ubar, S.Devadze, J.Raik, A Jutman. Parallel Fault Backtracing for Calculation of Fault Coverage. 13th ASPDAC, Seoul, Korea, 2008, pp. 667-672.

[17] 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.

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

[19] K.-T. Cheng, H.-C. Chen, “Classification and Identification of Nonrobust Untestable Path Delay Faults,” *IEEE Trans. On CAD of IC and Systems*, Vol.15, No.8, 1996, pp. 845-853.

[20] 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, pp. 660-666.

Publication VII

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

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 based on the fault reasoning (deductive [2], concurrent [3] and differential simulation [4]) is proved to be as well 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].

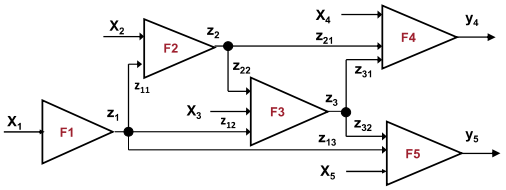


Fig.1. Combinational circuit with five FFRs

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.

Let us 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 symbolic, 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.

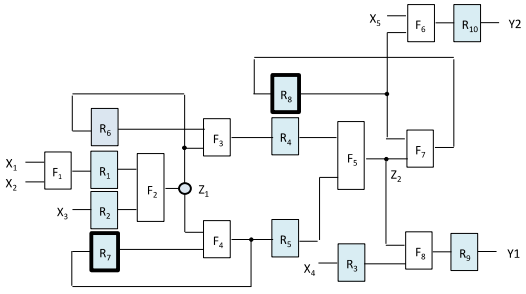


Fig.2. Sequential circuit

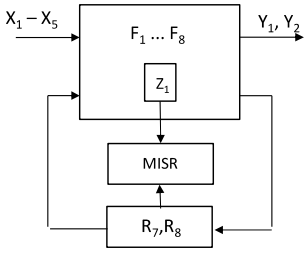


Fig.3. Sequential circuit with MISR

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):

$$\begin{aligned} Z_1 &= f_{z1}(X_1^{-1}, X_2^{-1}, X_3^{-1}) \\ R_7 &= f_{R7}(RES_7^{-2}, Z_1^{-1}) \\ R_8 &= f_{R8}(RES_8^{-2}, R_7^{-2}, Z_1^{-2}, Z_1^{-3}) \\ Y_1 &= f_{Y1}(R_7^{-2}, Z_1^{-2}, Z_1^{-3}, X_4^{-2}) \\ Y_2 &= f_{Y2}(X_5^{-1}, R_8^{-1}) \end{aligned} \qquad (3)$$

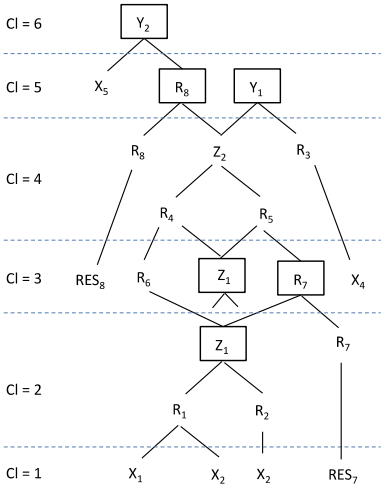


Fig.4. Simulation cycle of a single independent test

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 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 projects FP7 BASTION, H-2020 IMMORTAL, 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.

Publication IX

J. Kõusaar, S. Kostin, R. Ubar, S. Devadze, J. Raik, “Exact Parallel Critical Path Fault Tracing to Speed-Up Fault Simulation in Sequential Circuits”, *International Journal of Microelectronics and Computer Science*, vol. 9, pp. 9-18, October 2018.

Exact Parallel Critical Path Fault Tracing to Speed-Up Fault Simulation in Sequential Circuits

Jaak Kõusaar, Sergei Kostin, Raimund Ubar, Sergei Devadze, Jaan Raik

***Abstract*—We propose a new method to speed up stuck-at fault simulation for sequential circuits. The method combines exact parallel critical path tracing of faults, used so far only for combinational circuits, with traditional fault simulation in sequential circuits. For that purpose, formulas are developed for classification of faults into two classes: the faults eligible for parallel critical path tracing, and the faults which have to be simulated over the global feedbacks in the circuit by traditional methods. Combining these two approaches in fault simulation – the combinational and sequential simulation concepts – allows dramatic speed-up of fault simulation in sequential circuits, which is demonstrated by experimental results.**

***Index Terms*—sequential circuits, fault simulation, stuck-at-faults, critical path tracing**

I. Introduction

FAULT simulation in digital circuits is one of the most important tasks in the fields of test and fault tolerance. By fault simulation the fault coverage of testing can be calculated, different data structures like fault dictionaries for purposes of fault diagnosis can be created, the quality of fault tolerance can be evaluated, and many other design analysis related tasks can be solved. Whereas the fault simulators of combinational circuits are fast and efficient, the fault simulation in sequential circuits is very slow due to the need of sequential fault propagation analysis over the global feedback structures.

Therefore, accelerating of fault simulation procedures in sequential circuits would have a strong impact to all of the mentioned applications.

In the past, a lot of different concepts and methods have been proposed for fault simulation in combinational circuits based on the idea of parallel pattern single fault propagation (PPSFP) [1]. This concept has been combined with other sophisticated simulation techniques such as test detect [2], critical path tracing [3, 4], stem region [5] and dominator concept [4, 6]. These techniques have reduced further the simulation time.

Another trend based on the fault reasoning (deductive [7], concurrent [8] and differential simulation [9]) is proved to be as well 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.

J. Kõusaar, S. Kostin, R. Ubar, S. Devadze and J. Raik are with the Department of Computer Engineering, TTU, Ehitajate tee 5, 19086 Tallinn, Estonia (e-mails: jaak.kousaar@gmail.com, raiub@pld.ttu.ee, sergei.kostin@gmail.com, serega@pld.ttu.ee, jaan@pld.ttu.ee)

The original critical path tracing method [3, 4] eliminates explicit fault simulation for faults within Fan-out-Free Regions (FFR). However, the explicit simulation of faults at fan-outs is 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 [10]. However, the rule based strategy does not allow parallel analysis and rule check for 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 for a general class of X-faults, and in [13], additional gain from parallel computing to speed-up the method was achieved by implementing it in the multi-core computing environment. The main idea of the PPECPT method is in compiling a dedicated compact computing model through the circuit topology analysis, which allows exact critical path tracing in parallel for a bunch of test patterns throughout the full circuit and not only inside FFRs.

Unfortunately, for sequential circuits the critical path tracing of faults is not possible, due to the sequential (time related) dependence of signals in the circuit, which in general case leads to the need of critical path tracing over many clock cycles, and hence to the exponential grow of calculations needed.

In this paper, we develop a new method, which allows partial parallel critical path tracing of faults in sequential circuits, combined with traditional separate fault simulation and, as the result, the speed of fault simulation in sequential circuits can be drastically increased.

We consider in this paper the class of stuck-at-faults (SAF), however, the results can be extended also to other fault classes like conditional SAF, transition delays, and *X*-faults, in a similar way, as it was shown in [12] for the case of combinational circuits.

The rest of the paper is organized as follows. In Section 2, we present a short overview of the exact parallel critical path tracing in combinational circuits for analytical reasoning of faults. In Section 3, we show why the direct exact parallel critical path tracing of faults is not possible in sequential circuits. Section 4 describes the concept of combining

analytical critical path based fault reasoning with traditional methods of sequential fault simulation. Section 5 presents the algorithm of separating the faults into two parts, and describes the fault simulation method as a whole. In Section 6 we give the formulas of estimating the speed-up of the method and develop the higher bounds of speed-up. Section 7 presents experimental data, Section 8 provides a discussion on the results of experimental research, and Section 9 concludes the paper.

II. Overview of the Method of Exact Parallel Critical Path Tracing of Faults

Consider a combinational circuit as a network of fan-out free regions (FFRs) [14] where each of the FFRs can be represented as a Boolean function $y = F(x_1, x_2, \ldots x_n) = F(X)$, with $X = (x_1, x_2, \ldots x_n)$ as input vector.

The fault simulation for a FFR, according to the 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 as illustrated in Fig. 1.

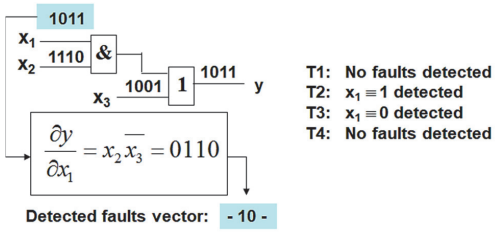
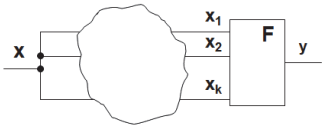


Fig. 1. Parallel fault simulation for an FFR at the gate level

In order to extend the parallel critical path tracing beyond an FFR described by a function $y = F(x_1,x_2,\ldots,x_k)$, where the variable x_i, $i = 1,2,\ldots,k$, represent the end nodes of paths which fan out from a joint node x, we can use the formula depicted in Fig. 2.



$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}),\ldots,(x_k \oplus \frac{\partial x_k}{\partial x}))$$

Fig. 2. Parallel fault simulation beyond the FFR

For generalization of this approach to any arbitrary network of FFRs, we can use the concept of Boolean differentials [15, 16], as shown in the following.

Consider the full Boolean differential for an FFR with a function $y = F(X)$ as

$$dy = y \oplus F((x_1 \oplus dx_1),\ldots(x_n \oplus dx_n)) = y \oplus F(X \oplus dX) \quad (1)$$

Here, by dx we denote the change of the value of x because of a fault at x, whereas $dy = 1$ corresponds to the case when an 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 [12] it was shown that from the expression (1), the following relationship can be derived:

$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}),\ldots,(x_n \oplus \frac{\partial x_n}{\partial x})) = y \oplus F(X \oplus \frac{\partial X}{\partial x})$$

or, in the vector form as

$$\frac{\partial y}{\partial x} = y \oplus F(X' \oplus \frac{\partial X'}{\partial x}, X'') \quad (2)$$

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

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 given convergent fan-out region by consecutive calculating of Boolean derivatives over the related FFR chains starting from x up to y. For that purpose, for each fan-out stem, which has convergence, the corresponding formulas like (2) should be constructed for all FFRs involved in the convergence. In the case of nested convergences, the formulas will have as well a nested structure. All these formulas will constitute a compiled partially ordered computation model for fault simulation, which can be composed by the topological analysis of the circuit [12].

Since the formulas of the compiled simulation model are Boolean, all computations on this model can be carried out in parallel for a bunch of test patterns. This computation procedure corresponds to parallel critical path tracing. The main advantage of this method is the possibility to calculate the full subset of faults, detectable by the simulated bunch of test patterns, by a single run through the compiled computation model.

III. The Problems Which Make Parallel Critical Path Tracing in Sequential Circuits Difficult

The described method for critical path tracing, both for single and parallel simulation of multiple patterns is possible only for combinational circuits, however, taking into account that the higher is the number of converging fan-out stems, the more complex is the computation model, and the slower will be the simulation run.

Consider in the following the reasons why exact critical path tracing of faults is practically impossible in sequential circuits.

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 Chapters 2, and which has been developed for using in 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 it will be shown, can be resolved if there will be a possibility to detect the fault in the first occasion when it has propagated up to any observable point without cycling in global loops.

There are two reasons why the same fault can be propagated to the same component of the circuit along different paths in different time frames:

(1) the case of a global feedback loop which includes a component to which the same fault can have influence via different time frames;

(2) the case of a sequential re-converging fan-out, where the fault may propagate to the same component from the same fan-out stem via different time frames.

The first case (1) is illustrated in Fig. 3. The circuit represented in Fig. 3 consists of three registers and four combinational blocks connected by busses. Assume, there is a fault Q in the sequential circuit on one of the outputs of the register R_2. Assume as well that the fault is propagating at the given test sequence to the primary output Y of the circuit by two successive test patterns. The first pattern propagates the fault in the clock cycle t-1 through combinational blocks F_3, F_4 and F_2 (black bold lines) to the register R_3, and the second test pattern propagates the erroneous signal from the register R_3 in the next clock cycle t via block F_4 to the primary output Y (red bold lines).

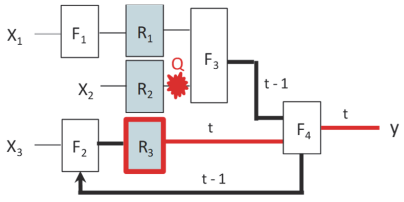


Fig. 3. Critical path tracing of a fault in a sequential circuit (the case (1))

To better understand the mechanism of critical path tracing of faults in the sequential circuit, let us unroll the sequence of two test patterns into two time frames t-1 and t of the iterative logic array of the circuit presented in Fig. 4.

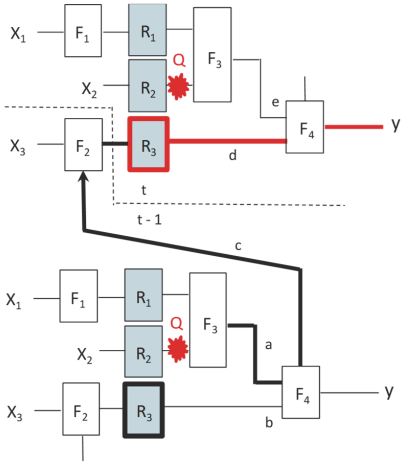


Fig. 4. Critical path tracing of a fault in a sequential circuit over two successive time frames (the case (1))

In the iterative logic array in Fig.4, the single fault Q is propagating, and in such a way its impact is re-converging on the inputs of the block F4 via two different paths: the path (Q, a, c, d) activated in the time frame t-1, and the path (Q, e) activated in time frame t.

Note that the critical path tracing of faults at the given test pattern is based on reasoning of the simulated correct signals in the circuit produced by the pattern.

Using only the correct signals makes it possible to determine in parallel all the faults which may propagate along the activated critical paths to the observable primary outputs. In Fig.4, there is activated a two cycle critical path along lines a, c (during cycle t-1), and d (during cycle t). The conditions of propagating the faults from a to c are determined by signals on the bus b at the time t-1, and the conditions of propagating the faults from d to the output Y are determined by signals on the bus e at the time t.

The critical path tracing of faults is carried out from the observable primary outputs towards the inputs. Starting from the output Y, the first step of the critical path tracing is to determine if the signals on the bus e are proper to propagate the faults from the line d through the block F_4 to the output Y. Since the fault Q under investigation should be considered in all time frames of the iterative array, the fault propagation conditions of the bus e are essentially depending also on the impact of the fault Q. However, this contradicts to the mechanism of critical path tracing of faults which must be carried out on the basis of using only the correct signals (also on the outputs of the register R_2).

The second case (2) of a sequential re-converging fan-out is illustrated in Fig. 5. In this case, the single fault Q is propagating to a converging point not along a global feedback, but rather along two branching paths activated at different time frames.

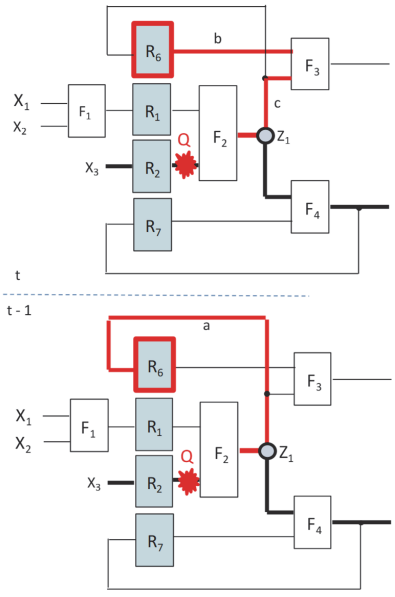


Fig. 5. Critical path tracing of a fault in a sequential circuit over two successive time frames (the case (2))

Assume, there is a fault Q in the sequential circuit on the output of the register R_1. Let us unroll the sequence of two test patterns into two time frames t-1 and t as shown in Fig. 5. The first branch in Fig. 5 consists of the path (Q, a, R_6) activated at time t-1, and of the path (R_6, b, F_3), activated at time t. The second re-converging branch is formed by the path (F_2, c, F_3), which is activated at time t. Both branches, propagating the impact of the same fault Q, are converging on the inputs of the block F_3 at the same clock cycle t.

For this case, we can show in the similar way as in the case (1) that the critical path tracing, using only the correct signals in the circuit, is not possible. As an example, in Fig. 5, in the time frame t, the faults on the line c cannot be back-traced, since the signal b at this time frame is not correct.

In such a way, the examples discussed on the basis of Fig. 4 and Fig. 5 demonstrate that in sequential circuits, each fault must be simulated separately, and the classical fault independent critical path tracing of faults used in combinational circuits is not possible.

In the next Section it will be shown how the critical path tracing method can be used still at least for one part of faults of sequential circuits which will help to considerably decrease the total amount of time needed for fault simulation in sequential circuits.

IV. Converting Sequential Fault Simulation into Combinational Critical Path Fault Tracing

Consider a sequential circuit in Fig. 6, partitioned into the combinational part, consisting of sub-circuits A, B, and C with related fault sets S_A, S_B, S_C, and a sequential part consisting of flip-flops FF. Let $S = S_A \cup S_B \cup S_C$ be the full set of faults to be simulated in the circuit.

The faults in S_A will always propagate directly to the primary output OUT_A, and never to FFs. Hence, the faults in S_A can be handled in all time frames of the given test sequence independently and they can be simulated by the parallel critical path tracing (PCPT) approach. On the contrary, the effect of the faults in S_B can never be observed directly at OUT_A without propagating one or more times through the loop of the feedback FF, and hence, these faults are tending to be self-masked (similarly to the example in Fig.3 and Fig.4). Consequently, the faults S_B cannot be analysed using PCPT, and they have to be processed by slow sequential fault simulation (SSFS) used traditionally in sequential circuits.

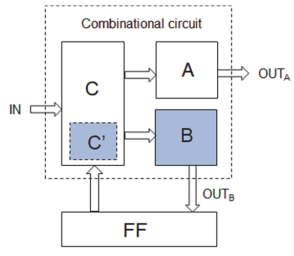


Fig. 6. A sequential circuit partitioned according to fault types

The faults in S_C represent a special case. When a test sequence is applied to the circuit, the same faults in S_C may sometimes propagate directly to the primary output OUT_A, whereas in other times they may propagate first to OUT_B, and only then, after in the feedback loop, they may be observed at the primary output OUT_A. Let the faults in S_C, which propagate first to the pseudo-output OUT_B, form a subset $S_{C'} \subseteq S_C$. The described ambivalence of the faults in S_C makes of them the key problem of the new fault simulation method.

The set of faults which propagate directly to the primary output OUT_A, without circulating in the feedback loop, can be represented as

$$S(A) = S_A \cup (S_C - S_{C'})$$

These faults use for propagating to the observable output OUT_A a single time frame, and hence, can be simulated in the same way as in the combinational circuits using the very fast PCPT simulation approach. Note, the PCPT is carried out using the correct signals of the circuit, which is the prerequisite of the possibility of parallel analysing concurrently all faults in the circuit.

The rest of faults

$$S(B) = S_B \cup S_{C'} = S - S(A)$$

need for propagating to the observable output OUT_A at least two (or more) time frames. This means that each fault, starting from the second time frame in its propagation trace up to OUT_A, will distort the states of signals in the circuit in all time frames in its own way, making the concurrent analysis of different faults not possible. Hence, all the faults in $S(B)$ have to be simulated separately by conventional methods used for sequential circuits, which is a very slow procedure.

Since the conventional fault simulators for sequential circuits are targeting by this slow approach all faults S in the circuit, it would be promising to extract from S the subset of faults $S(A)$ which can be simulated using fast PCPT. In this way, it would be possible to speed up dramatically the full fault simulation procedure.

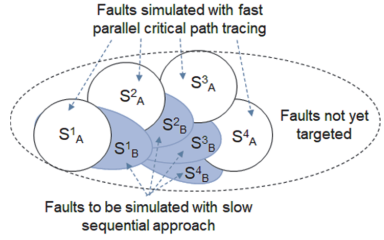


Fig. 7. Simulation based fault classification

We propose here the following simulation based method for classification of all faults into the two subsets $S(A)$ and $S(B)$ where $S(A) \cap S(B) = \varnothing$.

The difficulty is related to the fault set $S_{C'}$, because the content of $S_{C'}$ is strongly dependent on the test sequence. In other words, it is not possible to predefine $S_{C'}$ before fault simulation, and the content of $S_{C'}$ can be defined only "on-line" during the process of fault simulation. This additional payload may slow down the full procedure.

The method, illustrated in Fig.7, is based on applying critical path tracing for the patterns in sequence, one by one, separately for the outputs OUT_A and OUT_B. As the result of the method, we will hit two targets: first, we find immediately using critical path tracing a part of detected faults $S(A)$, and second, we extract the subset of faults $S(B)$ which will need separate slow fault simulation.

We start from the first pattern of the test sequence, and calculate the sets S^1_A and S^1_B of faults detected on the outputs OUT_A and OUT_B, respectively. The faults of S^1_A can be included immediately into $S(A)$ of detected faults by this first pattern, $S(A_1) = S^1_A$, whereas the faults S^1_B we include into $S(B_1)$.

Next, we find for the second pattern of the test sequence the sets of detected faults S^2_A and S^2_B. The sets $S(A)$ and $S(B)$ can be then updated in the following way:

$$S(A) = S(A_2) = S(A_1) \cup (S^2_A - S(B_1)),$$
$$S(B) = S(B_2) = S(B_1) \cup (S^2_B - S(A_2)).$$

Following now the example in Fig. 5, we can express the state of the fault simulation in general case after the k-th pattern of the test sequence:

$$S(A) = S(A_k) = S(A_{k-1}) \cup (S^k_A - S(B_{k-1})) \quad (3)$$
$$S(B) = S(B_k) = S(B_{k-1}) \cup (S^k_B - S(A_k)). \quad (4)$$

The procedure ends if the k-th pattern will be the last pattern of the test sequence. $S(A)$ will represent the set of faults simulated and detected very fast by the critical path tracing method, and $S(B)$ will represent the faults which need additional fault simulation by any conventional fault simulator for sequential circuits.

V. Combining Parallel Critical Path Tracing with Sequential Fault Simulation

In the previous section, we considered a test for a sequential circuit as a single test sequence where all test patterns, consisting of primary input patterns and pseudo-input patterns (the output values of flip-flops), are strongly depending on each other due to the feedback loop. This fact allowed us to exploit the power of critical path tracing only for single patterns. In Section II we described the method of parallel critical path tracing concurrently for many patterns, which was developed for combinational circuits. This parallelism is possible due to the independency of test patterns in case of combinational circuits.

In the following, we develop a method for parallel critical path tracing also for sequential circuits, which exploits the independences between the test segments in the full test sequence. We assume that each segment will consist of the initialisation of the flip-flops, and the subsequent patterns will serve for fault sensitization and fault propagation to the primary output OUT_A.

Consider a test sequence consisting of k test segments, where each i-th test segment consists of m_i test patterns $TS_i = (T_{i,1}, T_{i,1}, \ldots T_{i,mi})$. Since all test segments are independent, then also the first patterns $(T_{1,1}, T_{2,1}, \ldots T_{k,1})$ in all k segments, and in a similar way, the second patterns $(T_{1,2}, T_{2,2}, \ldots T_{k,2})$ in all k segments, etc., can be considered together as a set of m packages $\{TPP_i\}$ of independent test patterns. The lengths of the packages, in general case, may be different, and the number of packages m is equal to the number of the patterns in the longest test sequence.

An example of converting the initial test sequence into a set of packages of independent test patterns is depicted in Fig. 8.

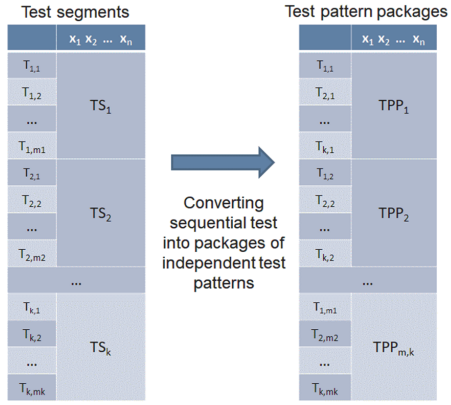


Fig. 8. Converting the set of test segments into the set of independent test pattern packages

For each package $TPP_j = (T_{1,j}, T_{2,j}, \ldots T_{k,j})$, $j = 1,2,\ldots, m$, of test patterns, parallel critical path tracing can be applied concurrently for all patterns in the package. As the result of this procedure, for each test pattern $T_{i,j} \in TPP_j$, the fault sets $S^{i,j}_A$ and $S^{i,j}_B$ will be calculated, and the detected fault sets (fault covers) will be calculated in the following way.

$$S^j_A = S^{1,j}_A \cup S^{2,j}_A \cup \ldots \cup S^{k,j}_A$$
$$S^j_B = S^{1,ja}_B \cup S^{2,j}_B \cup \ldots \cup S^{k,j}_B$$

An example of a fault table created by parallel critical path tracing for the given set of k packages of independent test patterns is depicted in Fig. 9.

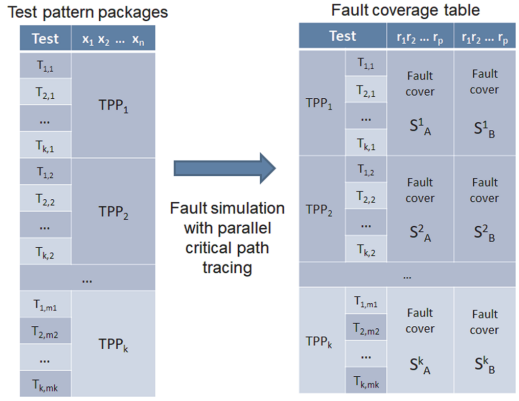


Fig. 9. Results of parallel critical path tracing of sequential test

Based on the fault sets S^j_A and S^j_B calculated by critical path tracing, the sets of faults $S(A_k)$ and $S(B_k)$ can be calculated similarly to the formulas (3) and (4), respectively.

The full procedure of the method of combining parallel critical path tracing with sequential faults simulation for sequential circuits can be presented as follows.

Algorithm 1

1: Convert the set of test segments into test packages of independent test patterns (see Fig. 8)
2: **for** each test package TPP_j
3: Apply parallel critical path tracing to calculate the fault sets S^j_A and S^j_B
4: **end for**
5: **for** k = 1,2,..,m (m is the number of test packages, $S(A_0) = S(B_0) = \varnothing$)
6: $S(A_k) = S(A_{k-1}) \cup (S^k_A - S(B_{k-1}))$
7: $S(B_k) = S(B_{k-1}) \cup (S^k_B - S(A_k))$
8: **end for**
9: **return** $S(A) = S(A_m)$, $S(B) = S(B_m)$,

The set of S(A) represents the faults detectable by the given test, calculated by parallel critical path tracing (PCPT). The set of S(B) represents the faults whose detectability is not possible to determine by PCPT. The faults in S(B) have to be simulated by conventional fault simulation methods used for sequential circuits.

VI. Estimating the Speed-up of the Proposed Method

Let us calculate now the characteristics of the timing analysis and of possible speed-up of the proposed method. Let us use the following notations:

n – is the number of faults in the circuit;
n_B – is the number of faults in S(B) to be simulated by slow sequential fault simulator;
t_{anal} – the total time needed for critical path tracing and fault analysis consisting of two parts $t_{anal} = t_{CP} + t_{CL}$, where
t_{CP} – time needed for critical path tracing, and
t_{CL} – time needed for fault classification.
t_{OLD} – time needed for fault simulation of all faults in a sequential circuit by conventional simulation;
t_{seq} – the average time of sequential simulation of a fault $t_{seq} = t_{OLD}/n$.

The total time needed for the proposed method, which combines the critical path tracing with conventional sequential fault simulation can be calculated as

$$t_{NEW} = t_{anal} + (t_{seq} \times n_B) = t_{CP} + t_{CL} + (t_{OLD} \times n_B) \qquad (5)$$

The speed-up of using the proposed method compared to the sequential approach can be characterized as

$$\frac{t_{OLD}}{t_{NEW}} = \frac{t_{seq} \times n}{t_{anal} + (t_{seq} \times n_B)}$$

Denote $p = t_{anal} / (t_{seq} \times n_B)$.

Since

$$t_{anal} << t_{seq} \times n_B ,$$

it would be reasonable to calculate the higher limit of speed-up when $p \to 0$:

$$\lim_{p \to 0} \frac{t_{OLD}}{t_{NEW}} = \lim_{p \to 0} \frac{n \times t_{seq}}{n_B \times t_{seq}} = \frac{n}{n_B} . \qquad (6)$$

From (6) we see that the speed-up of the proposed method is substantially depending on the number of faults n_B in S(B), which cannot be simulated by critical path tracing. Note, the content of S(B), and the speed-up effect of the proposed method depends also on the given test sequence to be fault simulated.

Since the set of S(B) can be calculated by parallel critical path tracing of faults separately by each pattern, the procedure is very fast compared to the traditional methods of sequential fault simulation.

VII. Experimental Data

The goal of the experimental research was to measure and investigate the speed-up potentials of the proposed new method for fault simulation in sequential circuits compared to the conventional fault simulation. The comparison was carried out for a representative selection of 27 circuits with different complexities (numbers of possible faults) of two benchmark families ISCAS'89 [17] (16 circuits) and ITC'99 [18] (11 circuits). The numbers of SAF faults in the circuits ranged from 614 up to 129422

Experiments were run on Intel i7-6700 4 cores 3.4 GHz, 32 GB RAM 2133 MHz, Windows 10, 64-bit.

Table I presents the main characteristic data of the circuits used in the experiments. We use the following notations: the number of inputs (#in), the number of outputs (#out), the number of flip-flops (#FF), the ratio between the numbers of observable primary outputs and not directly observable flip-flops #Out/#FF, which characterizes the sequential complexity of the circuit, the number of gates (#gates) and the number of possible stuck-at-faults (#faults).

Table II presents data of the experimental research, where the benchmark circuits are ordered according to their increasing complexity (the numbers of faults). We applied to each benchmark circuit a sequential test consisting of 32 independent test segments, where each segment was created of a sequence of 50 random input patterns. We assumed that before each test sequence the circuit flip-flops can be initialized (reset). The total length of the full test sequence was 1600 test patterns.

In columns 4-6, the test sequence is characterized for all circuits by two types of fault coverages s(A) and s(B), which mean the percentages of faults in the sets S(A) and S(B), respectively, in relation to the full set of faults given in column 3. The sum s(A) + s(B) in column 6 characterizes the sets of faults which propagate to the primary outputs and flip-flops, respectively. Note, the faults in S(A) are detected directly by the critical path tracing, but the faults in S(B) are those which need additional sequential fault simulation to determine if they propagate as well via feedback loops to the primary outputs.

All simulation time costs are given in seconds. The time costs t_{CP} and t_{CL} in columns 7 and 8 characterize the time needed for critical path tracing and the time needed for fault classification, respectively. Hence, the time $t_{CP} + t_{CL}$ represents the full cost of the parallel fault simulation of faults by critical path tracing. The total time cost t_{NEW} of fault simulation by the new method proposed in the paper, where the parallel critical path tracing and the traditional sequential fault simulation are combined, is depicted in column 9.

Columns 10 and 11 present the time costs of the baseline methods – the traditional sequential fault simulation. Here, two approaches are considered – simulation with fault dropping [14] (t_{OLD_FD}), and simulation with non-fault dropping (t_{OLD_NFD}). In column 12 the advantage (the gains in time cost reduction) of the proposed method is presented, and in column 13, for

comparison, the higher bounds of the possible gains, calculated by the formula (6), are depicted.

Note, the target of this research was not to generate test patterns with highest fault coverage, rather, the goal of experiments was to investigate the speed-up of fault simulation compared with the baseline, for any test sequence generated and supposed to be the objective of fault simulation. In the current case, randomly generated test sequences were selected with equal test length for all circuits.

TABLE I
CHARACTERISTICS OF THE BENCHMARK CIRCUITS USED IN EXPERIMENTS

Circuit	#in	#out	#FF	#Out/#FF	#gates	#faults
s349	9	11	15	0.7	161	614
s386	7	7	6	1.2	159	744
s510	19	7	6	1.2	211	900
s526	3	6	21	0.3	193	984
s641	35	24	19	1.3	379	1164
s713	35	23	19	1.2	393	1266
s953	16	23	29	0.8	395	1720
s1423	17	5	74	0.1	657	2610
s1494	8	19	6	3.2	647	2864
s5378	35	49	179	0.3	2779	9122
s9234	19	22	228	0.1	5597	16756
s13207	31	121	669	0.2	7951	24882
s15850	14	87	597	0.1	9772	29682
s35932	35	320	1728	0.2	16065	65248
s38417	28	106	1636	0.1	22179	69662
s38584	12	278	1452	0.2	19253	72346
b04	11	8	66	0.1	652	2836
b05	1	36	34	1.1	927	3952
b07	1	8	49	0.2	383	1712
b08	9	4	21	0.2	149	706
b10	11	6	17	0.4	172	806
b11	7	6	31	0.2	726	2894
b12	5	6	121	0.0	944	4426
b13	10	10	53	0.2	289	1350
b14	32	54	245	0.2	9767	38982
b15	36	70	449	0.2	8367	36496
b17	37	97	1415	0.1	30777	129422

VIII. DISCUSSION OF THE EXPERIMENTAL RESULTS

The test sequences were analysed twice. First, we calculated by the proposed method of parallel critical path tracing the sets of faults $\{S^k_A\}$ which propagated to the primary outputs OUT_A, and the sets of faults $\{S^k_B\}$ which propagated to the pseudo-outputs OUT_B (see Fig. 6 and Fig. 7). The time cost of this procedure was measured as t_{CP}. Then we classified the sets of faults into two subsets $S(A) \subset \{S^k_A\}$ and $S(B) \subset \{S^k_B\}$. The subset $S(A)$ included the faults which could be declared already as detected faults, as the result of critical path tracing method, whereas the subset of faults $S(B)$ included those faults which had to be simulated by a conventional simulator of sequential circuits. The time cost of fault classification procedure was measured as t_{CL}.

The experimental results of the gain in speed-up of the new method, compared with the baseline method, in column 12, and the higher bounds for the gain, are illustrated by the charts in Fig. 10. The new method outperforms the baseline method in a broad range of up to 5 times (in average 2 times), except of only very small circuits (less than 200 gates).

On the other hand, we see that the experimental results and the theoretically calculated bounds, using the formula (6), are very close in case of small circuits. In case of larger circuits, they nearly match even exactly. This gives an excellent possibility to predict by a simple calculation of the formula (6) the expected speed-up of fault simulation by the new proposed method.

TABLE II
EXPERIMENTAL RESULTS OF COMPARING THE PROPOSED METHOD VS. CONVENTIONAL FAULT SIMULATION METHODS FOR SEQUEMTIAL CIRCUITS

No	Benchmark circuits s - Iscas'89 b - ITC'99	Complexity of circuits (# faults)	Fault coverage of the simulated test sequence (%)			Time costs for critical path tracing of faults (s)		Total time costs for fault simulation (s)			Results	
								t_{NEW}	t_{OLD}		Gain t_{OLD_NFD} / t_{NEW}	Higher bound
			s(A)	s(B)	S(A)+S(B)	t_{CP}	t_{CL}		t_{OLD_FD}	t_{OLD_NFD}		
1	2	3	4	5	6	7	8	9	10	11	12	13
1	s349	614	8.3	88.1	96.4	0.10	0.01	0.5	0.03	0.44	0.89	1.13
2	b08	706	2.3	96.0	98.3	0.11	0.01	0.6	0.05	0.51	0.84	1.04
3	s386	744	36.0	36.0	72.0	0.12	0.01	0.3	0.10	0.49	1.61	2.78
4	b10	806	3.0	89.5	92.4	0.10	0.01	0.7	0.08	0.65	0.93	1.12
5	s510	900	28.2	65.1	93.3	0.11	0.02	0.6	0.12	0.73	1.21	1.54
6	s526	984	3.3	32.3	35.6	0.10	0.01	0.4	0.25	0.80	2.14	3.09
7	s641	1164	50.0	37.2	87.2	0.12	0.03	0.7	0.16	1.56	2.15	2.69
8	s713	1266	43.8	37.6	81.4	0.12	0.03	0.8	0.23	1.75	2.17	2.66
9	b13	1350	2.5	62.0	64.5	0.12	0.02	1.2	0.41	1.72	1.43	1.61
10	b07	1712	1.1	79.8	80.8	0.14	0.05	2.3	0.47	2.70	1.15	1.25
11	s953	1720	8.0	86.3	94.4	0.16	0.03	2.5	0.31	2.73	1.07	1.16
12	s1423	2610	2.3	68.5	70.8	0.16	0.05	4.6	1.04	6.38	1.39	1.46
13	b04	2836	1.1	89.9	91.0	0.18	0.15	7.4	0.57	7.84	1.06	1.11
14	s1494	2864	37.9	20.2	58.1	0.15	0.07	1.5	1.38	6.28	4.22	4.95
15	b11	2894	0.8	67.3	68.2	0.16	0.04	5.3	1.58	7.60	1.43	1.48
16	b05	3952	2.2	32.2	34.4	0.21	0.16	4.5	4.22	12.99	2.86	3.11
17	b12	4426	0.5	38.4	38.9	0.20	0.07	6.3	4.25	15.83	2.50	2.61
18	s5378	9122	45.7	26.2	71.9	0.36	0.24	22.9	11.00	85.21	3.72	3.82
19	s9234	16756	2.1	30.1	32.2	0.58	0.36	91.1	105.00	300.03	3.29	3.33
20	s13207	24882	6.3	45.7	52.0	0.84	0.51	306.2	139.00	667.20	2.18	2.19
21	s15850	29682	5.1	32.5	37.6	1.17	0.60	309.6	250.00	946.45	3.06	3.08
22	b15	36496	0.6	49.7	50.3	5.35	0.65	521.2	228.60	1036.21	1.99	2.01
23	b14	38982	0.5	74.6	75.1	3.48	0.73	919.3	258.80	1226.38	1.33	1.34
24	s35932	65248	8.8	75.6	84.4	1.51	0.95	2827.0	367.00	3734.21	1.32	1.32
25	s38417	69662	1.9	50.7	52.6	2.44	1.35	2516.8	1189.00	4956.59	1.97	1.97
26	s38584	72346	8.0	69.9	77.9	2.27	1.48	3255.9	704.00	4650.52	1.43	1.43
27	b17	129422	0.3	25.0	25.3	15.42	1.96	3226.2	2856.20	12857.76	3.99	4.01
Average						1.33	0.35	519.9	226.81	1130.80	2.0	2.2

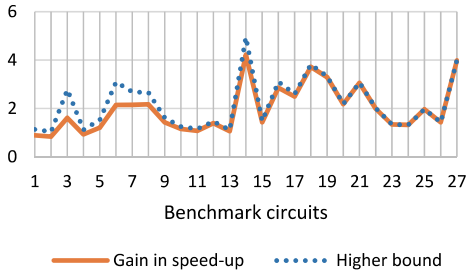


Fig. 10. Comparison of the speed-up of simulation with higher bound

Fig. 11 illustrates the dependence of the gain in speed-up of simulation on the complexity of circuits (the number of faults). To smooth the irregularities of the data for different circuits on the X-axis as shown in Fig. 10, and to take into account the trends of changing the comparable data, we compare the cumulative gain and the cumulative number of faults over the full set of benchmark circuits used in the experiments. The diagrams in Fig. 11 show that the trend of the gain is nearly linear over the full set of circuits ranked in order of complexity. We see from Fig. 11 that the linearity trend continues also in the second part of the ranked list of circuits (from 19 to 27) where the complexity starts to increase very rapidly.

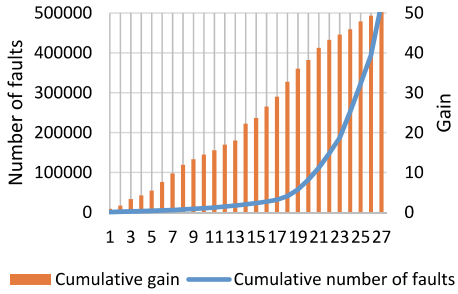


Fig. 11. Dependence of speed-up of simulation on the complexity of circuits

This finding allows to claim that the proposed method is well scalable, since the method starts to be in this comparison more efficient in case when the circuits' complexity will increase.

The cumulating functions were used in the graphics in Fig. 11 to smooth the anomalies of the gain curve in Fig. 10, and the nonlinearity of spreading the number of faults on the X-axis.

This anomaly can be explained by the two charts in Fig. 12, which illustrate the impact of the feedback intensities (the share of the faults propagating over the feedback loops) and the gain in speed-up of simulation for the full range of benchmark circuits. To characterize the feedback intensities, we use the values of s(B) in Table II, which represent the percentage of faults, which do not propagate directly to the primary output, but rather start propagating via feedback loops to the subsequent clock cycles. These faults represent the bottleneck of the proposed method, because they cannot be simulated by fast critical path tracing. In Fig. 12, we see that for all circuits where s(B) is high, the gain of the method is low, and opposite.

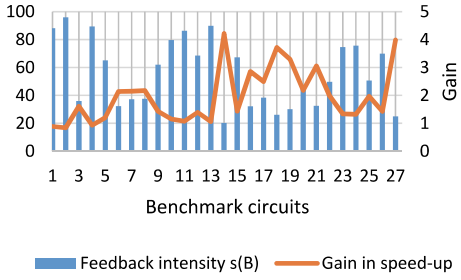


Fig. 12. Dependence of gain on the feedback level

The proposed method can be compared with two conventional fault simulation methods for sequential circuits as baselines: simulation with and without fault dropping, with time costs depicted in Table II in columns 10 and 11, respectively. The comparison results for the proposed method are shown also in Fig. 13. Here we see, that the proposed method outperforms well the traditional fault simulation without fault dropping, but loses in speed to the simulation with fault dropping.

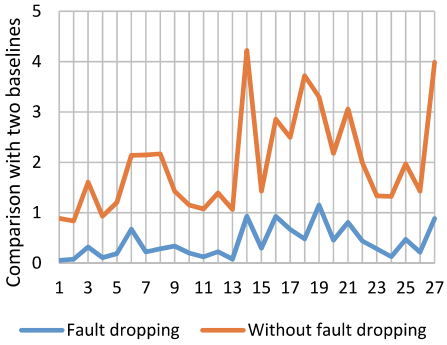


Fig. 13. Comparison of the speed of the proposed method with two baselines: sequential fault simulation with and without fault dropping

Note, however, that the fault dropping based fault simulation approach is able to evaluate only the fault coverage of the given test, whereas the simulation without fault dropping provides not only the fault coverage data, but also the data for fault diagnosis purposes, e.g. the data needed for creating fault tables and fault dictionaries.

On the other hand, the main core of the proposed method, the parallel critical path tracing of faults, is substantially tailored to get diagnostic information for each test pattern separately, in a similar way that the sequential simulation without fault dropping does. Hence, the comparison baseline for the proposed new method should be namely simulation without fault dropping, and it would be even unfear to compare the new method with simulation which uses fault dropping approach.

In Fig. 12 we saw that the high values of $s(B)$, which characterize the sequential level of the given circuit, act against the efficiency of the proposed method. The values of $s(B)$ will be high, when the ratio of the number of flip-flops and the number of primary outputs (#FF/#out) is very high. On the other

hand, we see also from Fig. 12, that if the value of $s(B)$ will reduce, then the speed of the proposed method will increase. This fact can motivate the redesign of sequential circuits for better observability to reduce the cost of testing, both, the test length and the time of fault simulation.

In Table III, an experiment is presented with 8 different designs of a circuit b17 with different numbers of flip-flops made observable by additional primary outputs. The row 1 corresponds to the original circuit b17 used in Table I and Table II, whereas the last row shows the extreme case where all flip-flops are directly observable. Here we see a dramatic speed-up of simulation and increasing of the gain, if the number of directly observable flip-flops will grow.

TABLE III
DEPENDENCE OF THE SPEED-UP GAIN ON THE OBSERVABILITY OF FLIP-FLOPS OF THE BENCHMARK CIRCUIT B17

Design No	s(A) %	s(B) %	Observable FFs, #	Time costs, s			Gain
				t_{CP}	t_{CL}	t_{NEW}	
1	0.3	25.0	0	15.80	2.01	3227	4
2	3.8	21.5	200	15.71	2.03	631	20
3	6.6	18.7	400	15.61	1.94	551	23
4	8.4	16.9	600	15.73	1.88	500	26
5	9.8	15.5	800	15.93	1.88	460	28
6	15.0	10.3	1000	15.70	2.04	312	41
7	23.6	1.7	1200	15.69	2	65	197
8	25.3	0.0	1415	15.74	2.03	18	724

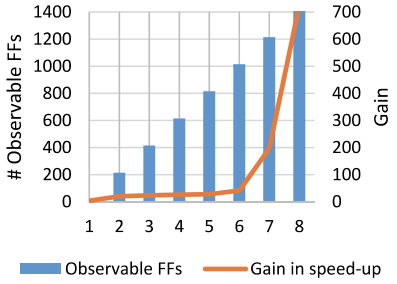


Fig. 14. Relationship between the speed-up gain and the observability of flip-flops of the benchmark circuit b17

As already mentioned, the results of the experimental research suggest a very fast procedure for prediction of the speed-up to be achieved by the new method compared to the conventional methods. Since

$$t_{anal} = t_{CP} + t_{CL} << t_{OLD}$$

the best way of such prediction would be to carry out Algorithm 1 for calculating the set S(B). The percentage of S(B) from the full set of faults will be a good criterion (higher bound according to the formula (6)) about the expected efficiency of using the proposed method.

IX. CONCLUSIONS

In this paper we have proposed a novel approach for fault simulation in sequential circuits, which allows to achieve considerable speed-up in simulation time compared to the conventional fault simulation methods. By experimental research, it was shown that for the pool of 27 benchmark circuits the average gain of speed-up was 2 times (in the best cases up to 5 times).

The high speed-up is achieved by integrating into the fault simulation process the exact parallel critical path tracing concept used so far only for fault simulation in combinational circuits.

The main novelties of the paper are as follows:

- we developed a novel method of separating the parallel critical path tracing into two parts, tracing to the primary outputs and tracing to the pseudo-outputs (to feedback loops);
- we developed a novel method for classification of the simulated faults into two classes: the faults directly detectable by combinational critical path tracing, and the faults to be simulated sequentially by traditional methods:
- the introduced fault classification approach allows to integrate combinational and sequential methods into a joint fault simulation procedure for sequential circuits.

We proposed also a very fast method for pre-evaluation of the applicability of the new fault simulation method, i.e. to predict the expected gain compared to the conventional methods. A dramatic speed-up can be expected if the size of s(B) is small. In case of the investigated benchmark circuits, the lowest value of $s(B)$ was 20% of the full set of faults, which provides speed-up gain 5 times. The size of $s(B)$ depends on the circuit characteristics, and also on the test sequence to be simulated.

ACKNOWLEDGEMENT

The work has been supported by EU's H2020 Twinning Action project TUTORIAL, Estonian institutional research grant IUT 19-1, and funded by Excellence Centre in IT in Estonia (EXCITE) project.

REFERENCES

[1] J.A. Waicukauski, et.al. Fault Simulation of Structured VLSI. VLSI Systems Design, Vol.6, No.12, pp.20-32, 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 CAD, Vol. 6, No. 5, pp.704-712, 1987.

[5] F. Maamari, J. Rajski, "A method of fault simulation based on stem region", IEEE Trans. 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. Int. Symp. on Defect and Fault Tolerance in VLSI Systems, Oct. 2005, pp.178-186.

[11] R. Ubar, S. Devadze, J. Raik, A. Jutman. Ultra Fast Parallel Fault Analysis on Structural BDDs. ETS, Freiburg, 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] M. Gorev, R. Ubar, S. Devadze. 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, 9-13 March, 2015.

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

[15] A. Thayse, M. Davio. “Boolean Differential Calculus and its Applications to Switching Theory”. IEEE Trans. Comput. V.C-22, No.4, pp. 409-420, 1973.

[16] Thayse, “Boolean Calculus of Differences”, Springer Verlag, 1981.

[17] F. Brglez, D. Bryan, K. Kominski, "Combinational Profiles of Sequential Benchmark Circuits", Int. Symp. on Circuits and Systems, 1989, pp.1929-1934.

[18] F. Corno, M.S. Reorda, G. Squillero, "RT-level ITC'99 Bench-marks and First ATPG Results", In Proc. Of the IEEE Design & Test of Computers, Vol. 17, No. 3, 2000, pp.44-53.

Jaak Kõusaar is a software engineer at Kuehne+Nagel and a PhD student at Tallinn University of Technology. He received MSc degree in 2013 from Tallinn University of Technology (sup Prof. Raimund Ubar). His scientific interests include diagnostics of digital systems and agent-oriented programming.

Sergei Kostin received his PhD degree at the Tallinn University of Technology in 2012 on topic “Built-in self-diagnosis”. He is currently working as a researcher in the laboratory of department of Computer Systems at Tallinn University Technology. His research interests include embedded testing and diagnosis as well as reliability of digital logic circuits in nanoscale designs.

Raimund Ubar is a professor at Tallinn University of Technology. He received PhD degree in 1971 from the Bauman Technical University in Moscow, and DSc degree in 1987 from the Latvian Academy of Sciences. His scientific interests include computer science, design for testability and diagnostics of technical systems. Ubar is a member of Estonian Academy of Sciences, life senior member of IEEE, and Golden Core member of IEEE Computer Society. He was awarded from the Estonian Government by White Cross Orden of III Class.

Sergei Devadze received the Ph.D. degree in computer engineering from TU Tallinn, Estonia in 2009, and currently holds the position of researcher in this university. His research interests embrace such topics as fault tolerance and fault management architectures of digital systems, fault simulation techniques, usage of chip-embedded instrumentation and IJTAG for board and system test. He is a co-author of over 70 scientific papers in the field of digital design and test.

Jaan Raik is a full professor at Tallinn University of Technology where he received his MSc and PhD degrees in 1997 and 2001, respectively. His research interests include verification, test and dependable design of digital systems. He is a co-author of more than 300 publications in the field.

Curriculum vitae

Personal data

Name:	Jaak Kõusaar
Date of birth:	August 10, 1976
Place of birth:	Tallinn
Citizenship:	Estonian

Contact data

Address:	15A Akadeemia St., 12618 Tallinn, Estonia
Phone:	+372 568 400 40
E-mail:	jaak@oyenetwork.com

Education

2013 – ...	Tallinn University of Technology	PhD studies
2012 – 2013	Tallinn University of Technology in Computer systems (*cum laude*)	Master's degree
2008 – 2012	Tallinn University of Technology in Computer systems (*cum laude*)	Bachelor's degree
1999 – 2004	Tallinn Pedagogical University sciences (unfinished)	Teacher of natural
1984 – 1995	Jakob Westholm Secondary School.	High school

Language competence

Estonian	Native speaker
English	C1
Russian	C1
Finnish	B2
German	A2

Professional employment

2018 – …	Warren.io, systems architect
2016 – 2018	OYE Network LTD, systems developer
2013 – 2016	Kühne+Nagel AS, systemresponsible
2008 – 2012	OV Omari OÜ, information systems developer
2002 – 2004	Tallinn Kuristiku Secondary School, teacher of chemistry and physics.
2000 – 2001	National Institute Of Chemical Physics And Biophysics, molecular genetics laboratory, assistant.
1998 – 1999	Espak Grupp, deputy head of department.
1996 – 1997	Eesti Statoil AS, customer service representative

Elulookirjeldus

Isikuandmed

Nimi:	Jaak Kõusaar
Sünniaeg:	10. august.1976
Sünnikoht:	Tallinn
Kodakondsus:	Eesti

Kontaktandmed

Aadress:	Akadeemia Tee 15A, 12618 Tallinn
Telefon:	+372 568 400 40
E-mail:	jaak@oyenetwork.com

Hariduskäik

2013 – ...	Tallinna Tehnikaülikool	Doktorantuur
2012 – 2013	Tallinna Tehnikaülikool	Magistrikraad, arvutisüsteemide eriala (*cum laude*)
2008 – 2012	Tallinna Tehnikaülikool	Bakalaureusekraad, arvutisüsteemide eriala (*cum laude*)
1999 – 2004	Tallinna Pedagoogikaülikool	Loodusteaduslike ainete õpetaja (lõpetamata)
1984 – 1995	Jakob Westholmi Gümnaasium.	Keskharidus

Keelteoskus

Eesti keel	Emakeel
Inglise keel	C1
Vene keel	C1
Soome keel	B2
Saksa keel	A2

Teenistuskäik

2018 – …	Warren.io, süsteemiarhitekt
2016 – 2018	OYE Network LTD, süsteemiinsener
2013 – 2016	Kühne+Nagel AS, systemresponsible
2008 – 2012	OV Omari OÜ, informatsioonisüsteemide arendaja
2002 – 2004	Tallinna Kuristiku Gümnaasium, keemia ja füüsika õpetaja
2000 – 2001	Keemilise ja Bioloogilise Füüsika Instituut, molekulaargeneetika labor, assistent.
1998 – 1999	Espak Grupp, osakonnajuhataja asetäitja.
1996 – 1997	Eesti Statoil AS, klienditeenindaja