
Resistive Bridging Faults

Defect-Oriented Modeling and Efficient Testing

Dissertation zur Erlangung des Doktorgrades der
Technischen Fakultät der Albert-Ludwigs-Universität
Freiburg im Breisgau



vorgelegt von
PIET ENGELKE

Januar 2009

Dekan: *Prof. Dr. Hans Zappe*
Albert-Ludwigs-Universität, Freiburg im Breisgau

Erstreferent: *Prof. Dr. Bernd Becker*
Albert-Ludwigs-Universität, Freiburg im Breisgau

Zweitreferent: *Dr. Michel Renovell*
Directeur de Recherche CNRS
Laboratoire d'Informatique, de Robotique et de
Microélectronique de Montpellier, Montpellier

Datum der Promotion: 27. März 2009

Zusammenfassung

Resistive Bridging Faults – Defektorientierte Modellierung und effizienter Test

Die vorliegende Dissertation beschäftigt sich mit defektorientiertem Test widerstandsbehafteter Kurzschlussdefekte. Die Auswirkung dieser Defekte auf das Verhalten digitaler Logikschaltungen wird durch das von Renovell et al. in [152, 154] vorgestellte Fehlermodell für widerstandsbehaftete Brückenfehler (engl. resistive bridging fault (RBF) model) beschrieben. Das Modell hat Gültigkeit für Schaltungen in komplementärer Metall-Oxid-Halbleitertechnologie (CMOS) und ist auf statischen Spannungstest ausgerichtet. In dieser Arbeit wird das Modell von Renovell et al. in mehrerer Hinsicht erweitert. Dadurch kann die Testbarkeit widerstandsbehafteter Kurzschlussdefekte signifikant verbessert werden. Die Erweiterungen betreffen zum einen die Fehlermodellierung als solche, wodurch das Einsatzspektrum des Modells deutlich vergrößert wird. Zum anderen wird auf die Entwicklung leistungsfähiger Algorithmen und Programme eingegangen. Mit Hilfe der hier diskutierten Programme ist es möglich, widerstandsbehaftete Kurzschlussdefekte in Schaltkreisen von praktisch relevanter Größe effizient zu testen. Des Weiteren demonstrieren umfassende Experimente, wie das in dieser Arbeit erweiterte Fehlermodell für widerstandsbehaftete Brückenfehler gewinnbringend in weiten Bereichen des Schaltungstests eingesetzt werden kann.

Die hier vorgestellten Erweiterungen des Fehlermodells für widerstandsbehaftete Brückenfehler betreffen unterschiedliche Aspekte. Es wird beschrieben, wie die elektrische Komponente des Modells an die Eigenschaften zweier aktueller CMOS-Prozesstechnologien angepasst werden kann. Damit ist die Anwendung des Modells auf Schaltungen neuester Bauart möglich. Zugleich verdeutlicht dies, dass das Modell dank seiner Anpassungsfähigkeit mit der ständig fortschreitenden technischen Entwicklung Schritt halten kann. Weiterhin wird die elektrische Komponente des Modells so erweitert, dass sich die Einflüsse der Versorgungsspannung und der Umgebungstemperatur auf die Auswirkungen des Kurzschlusses exakt modellieren lassen. Dadurch ist es möglich, die Wirksamkeit von Teststrategien zu untersuchen, bei denen die Versorgungsspannung (engl. low-voltage testing) und die Umgebungstemperatur (engl. low-temperature testing) moduliert werden. Des Weiteren wird ausgeführt, wie sich das Modell um die Möglichkeit ergänzen lässt, neben dem Spannungstest auch die Entdeckbarkeit von Kurzschlussdefekten im Rahmen eines Ruhestromtests (engl. IDDQ testing) zu evaluieren. Schließlich erlauben es die hier vorgestellten neuartigen Metriken zur Berechnung der Fehlerüberdeckung, die verschiedenen Testverfahren für widerstandsbehaftete Kurzschlussdefekte exakt miteinander zu vergleichen.

Eine Grundvoraussetzung für den praktischen Einsatz eines Fehlermodells für widerstandsbehaftete Kurzschlussdefekte ist die Existenz leistungsfähiger Fehlersimulatoren und automatischer Testmustergeneratoren (ATPG), die auf Grundlage dieses Modells arbeiten. Die vorliegende Arbeit präsentiert mehrere effiziente Algorithmen dank derer sich solche Programme realisieren lassen. Es wird gezeigt, wie sich die sogenannte Sectioning-Methode (engl. sectioning technique), die von Shinogi et al. [174] für widerstandsbehaftete Brückenfehler entwickelt wurde, mit einem bitparallelen Fehlersimulationsverfahren kombinieren lässt. Diese Kombination kommt in dem in dieser Arbeit beschriebenen Simulator Simulator Utilizing Parallel Evaluation of Resistive Bridges (SUPERB) zum Einsatz. Dieser Fehlersimulator ist deutlich schneller als alle vergleichbaren Programme und erlaubt die Verarbeitung von Fehlerlisten realistischer Größe für Schaltungen mit mehreren Millionen Gattern. Auf dem Gebiet der Testmustererzeugung wird das Programm RBF-ATPG vorgestellt. Dieses ist das erste Programm seiner Art, welches in der Lage ist Testmustersätze zu erzeugen, die Kurzschlussdefekte für alle entdeckbaren Widerstandsbereiche testbar machen. RBF-ATPG vereint die Sectioning-Methode mit einem Algorithmus zur Testmustererzeugung, der auf einer Prozedur zur Lösung des Erfüllbarkeitsproblems der Aussagenlogik (SAT) basiert. Ausgehend von den mittels RBF-ATPG berechneten entdeckbaren Widerstandsbereichen ist es nun möglich, die exakte Fehlerüberdeckung für widerstandsbehaftete Brückenfehler zu ermitteln. Weiterhin werden auf formalem Wege Bedingungen hergeleitet, unter denen sich eine bestimmte Klasse von widerstandsbehafteten Brückenfehlern garantiert entdecken lässt. Testmustererzeugung für diese Klasse von Fehlern wird auf Grundlage dieser Bedingungen deutlich vereinfacht.

Um den Nutzen der Erweiterungen des Fehlermodells und der Programme SUPERB und RBF-ATPG zu unterstreichen, wird in der vorliegenden Arbeit eine Vielzahl von Experimenten durchgeführt. Diese untersuchen unter anderem die Wirksamkeit von low-voltage und low-temperature Tests. Auf Grundlage der gewonnenen analytischen Ergebnisse werden Richtlinien angegeben, welche es ermöglichen, die Vorzüge der jeweiligen Technik optimal auszunutzen. Weitere Experimenten stellen Spannungs- und Ruhestromtest gegenüber. Auch in diesem Bereich lassen sich dank der in dieser Arbeit beschriebenen Erweiterungen die optimalen Anwendungsgebiete für jede der beiden Techniken analytisch feststellen. Darüber hinaus wird ein Experiment dargelegt, welches die Auswirkungen zweier Verfahren für den Selbsttest (engl. built-in self test, BIST) auf die Entdeckbarkeit von Defekten untersucht. Mit Hilfe der hier gezeigten Vorgehensweise lässt sich allgemein für Selbsttestverfahren bewerten, inwieweit sie die Entdeckbarkeit von Defekten beeinflussen. Schlussendlich wird evaluiert wie häufig widerstandsbehaftete Kurzschlussdefekte sogenannte Doppelfehler (engl. double error) hervorrufen. Die hierbei gewonnenen Ergebnisse sind insbesondere für die Analyse von Techniken zur Verbesserung der Fehlertoleranz von großem Interesse.

Acknowledgements

This work could only be completed successfully with the help of several people – I highly appreciate their contribution.

Most notably, I would like to express my gratitude to my supervisor, Prof. Dr. Bernd Becker, for the invaluable support which he has provided me during the course of this research. He created the excellent environment which made working at the University of Freiburg a stimulating and enjoyable experience.

My special thanks are due to the external examiner of this thesis, Dr. Michel Renovell, for the numerous discussions which provided profound insight into the modeling of resistive shorts. Without his contributions my research would not have been possible.

I also wish to extend special thanks to Dr. Ilia Polian for the very fruitful collaboration and for checking the manuscript. His creativity and scientific curiosity heavily influenced the direction of my research.

Furthermore, I would like to thank all my colleagues from the computer architecture group. In particular, Matthew Lewis, who proof-read the manuscript with special diligence. Last but not least, I want to thank my family, my girlfriend, and all my friends for their encouragement and constant support.

Freiburg im Breisgau, March 2009

Piet Engelke

Contents

1	Preface	1
1.1	Contributions	2
1.2	Structure of the Thesis	3
2	Preliminaries	5
2.1	Digital Circuits	5
2.1.1	Combinational and Sequential Circuits	5
2.1.2	Implementation of Digital Circuits	9
2.1.3	Levels of Abstraction	12
2.2	Testing of Digital Circuits	13
2.2.1	General Terminology	14
2.2.2	Testing of Sequential Circuits	15
2.2.3	Stuck-At Faults	16
2.2.4	Built-In Self Test	17
I	Bridging Fault Models	19
3	Fundamentals of Bridging Faults	21
3.1	Basic Properties and Terminology	21
3.2	Bridging Fault Model Classification	23
3.3	Bridging Fault Extraction	25
4	Non-Resistive Bridging Fault Models	29
4.1	Simple Logic Models	30
4.1.1	Wired-Logic And Dominance-Behavior Models	30
4.1.2	The 4-way Model	32
4.2	Technology-Based Models	33
4.2.1	The Voting Model	35
4.2.2	The Biased Voting Model	36
4.3	Generalized Logic Models	38
4.3.1	The Unified Model	39
4.3.2	The Precise Test Generation Model	42
5	The Resistive Bridging Fault Model	45
5.1	Introduction of the Resistive Bridging Fault Model	46

5.2	Calculating Critical Resistances	49
5.2.1	General Framework	50
5.2.2	Technology-Specific Models	51
5.3	Analogue Detectability Intervals	59
5.4	Fault Coverage Metrics	60
5.5	Fault Effect Propagation	65
5.5.1	Interval-Based Technique	66
5.5.2	Sectioning Technique	68
5.5.3	Feedback-Bridging Faults	72
5.5.4	Occurrence of Double Errors	74
5.5.5	Double Observation of Two Successor Bridging Faults	76
6	Summary and Discussion of Part I	81
II	Applications of the Resistive Bridging Fault Model	83
7	SUPERB – A Resistive Bridging Fault Simulator	85
7.1	Efficient Simulation of Resistive Bridging Faults	86
7.1.1	Fault List Preprocessing and Data Storage	87
7.1.2	Fault Simulation Procedure	89
7.2	Experimental Results	92
7.3	Conclusions	97
	Experimental Data	98
8	RBF-ATPG – A Test Pattern Generator For Resistive Bridging Faults	103
8.1	ATPG for Resistive Bridging Faults	105
8.1.1	Example: Deriving Test Patterns for Circuit	106
8.1.2	ATPG Algorithm in Detail	109
8.2	Experimental Results	112
8.2.1	Evaluation of n -Detection and 4-way Test Vectors	117
8.2.2	Test Pattern Generation for Different Technology Models	119
8.3	Conclusions	121
	Experimental Data	122
9	Advanced Testing Methods	127
9.1	Low-Voltage and Low-Temperature Testing	128
9.1.1	Extensions of the Resistive Bridging Fault Model	130
9.1.2	Experimental Results	137
9.1.3	Coverage Loss by Low-Voltage Testing	143
9.2	Delta-IDDQ Testing	146
9.2.1	Extensions of the Resistive Bridging Fault Model	148
9.2.2	Delta-IDDQ Testing of Sequential Circuits	153
9.2.3	Experimental Results	155
9.3	Conclusions	158
	Experimental Data	159

10 Benchmarking BIST Techniques	171
10.1 Detection of Non-Target Defects by Deterministic Logic BIST	172
10.1.1 Deterministic Logic BIST with Bit-Flipping	173
10.1.2 Experimental Results	176
10.2 Non-Target Defect Coverage Impact of X-Masking	179
10.2.1 The X-Masking Logic	180
10.2.2 Experimental Results	183
10.3 Conclusions	188
Experimental Data	189
11 Summary and Discussion of Part II	193
Concluding Remarks	195
Author’s Publications	197
Bibliography	201
List of Algorithms	215
List of Figures	217
List of Tables	221

1 Preface

In today's world electronic devices are virtually omnipresent. The application of *integrated circuits* (IC) is no longer limited to only "traditional" computers. Almost unnoticed ICs also became the central component of many "ordinary" appliances such as dish washers and coffee machines. All this is possible thanks to constant miniaturization of the IC manufacturing technology. The downside of this impressive growth in complexity is that assuring IC quality becomes more and more difficult. Low numbers of *defective parts per million* (DPM) are critical for the commercial success of a product. In many application domains, IC quality and reliability is even vital for the safety of the customers. One of the key manufacturing steps, which screens defective parts and thus guarantees IC quality, is *test*. Traditionally testing of digital circuits relies on the stuck-at fault model. Yet, for many years it has been known that a lot of defects are not adequately represented by this fault model [5, 43, 66, 79, 109, 131]. Unfortunately, the deficiency of the stuck-at fault model is even intensified by the constant miniaturization of integrated circuits.

Defect-oriented or defect-based testing [109, 169] is a systematic methodology to derive accurate fault models which closely match the behavior of actual defects occurring in the IC manufacturing process. According to [163] a *defect* is defined as follows:

“Defects are undesired features in the silicon layer structure of an IC. They can take form of missing or extra pieces of material, as well as of random and systematic shifts in the outcome of the semiconductor process.”

The class of *local defects* which is provoked by extra or missing material is often referred to as *spot defects* [163]. The *inductive fault analysis* (IFA) developed by Shen et al. [172] can be used to extract the most likely spot defects from the design of an integrated circuit. Furthermore, this technique allows to inductively construct a fault model from the defects observed in the IC. By IFA and related techniques it could be proven that one particularly prominent class of spot defects are the so-called *shorts* [45, 51]. This type of defect creates an unwanted connection between two or more conducting elements of the circuit. The impact of shorts on circuit behavior is represented by the resistive bridging fault (RBF) model which has been proposed by Renovell et al. in [152, 154].

This doctoral thesis aims at extending the resistive bridging fault model and explores potential application domains.

1.1 Contributions

In the following we present our contributions to defect-oriented testing of resistive shorts. We significantly improved the testability of these defects by enhancing the resistive bridging fault model, devised by Renovell et al., in several aspects. On the one hand we evolved the model, thus extending its scope of application. On the other hand we developed efficient algorithms and high-performance tools. These tools enable testing of resistive shorts in circuits of practically relevant size. Our extensive experiments demonstrate the benefits of the resistive bridging fault model in numerous application domains.

Our contributions to the core of the resistive bridging fault model cover several aspects. We transferred the model's versatile electrical framework to two recent technology nodes. This enables application of the model to current integrated circuits. At the same time, we demonstrated that its versatility allows the model to keep pace with technological progress. Furthermore, we extended the electrical framework to account for the impact of variable operating temperature and power supply voltage during test application. Hence, we are able to analyze the effectivity of low-voltage and low-temperature testing strategies in the coverage of resistive shorts. Moreover, we expanded the capabilities of the resistive bridging fault model to quiescent current testing. Thus, we can now directly compare voltage and quiescent current testing within the same analytical frame. Several novel fault coverage metrics are introduced to accurately evaluate the various testing options for resistive shorts.

A refined fault model for resistive shorts can only be of value if there are tools implementing test generation and fault simulation for that model. For this purpose we developed several efficient algorithms. We combined the well-known sectioning technique, which maps resistive bridging faults to stuck-at faults, with a bit-parallel fault simulation engine. This algorithm is implemented in our novel, high-performance Simulator Utilizing Parallel Evaluation of Resistive Bridges (SUPERB). We demonstrated that the resistive bridging fault simulator SUPERB is substantially faster than competing state-of-the-art tools for the same fault model. SUPERB enables accurate resistive bridging fault simulation of multi-million gate designs for fault lists of realistic size. Furthermore, we considerably advanced test pattern generation for resistive bridging faults by our tool RBF-ATPG. This automatic test pattern generator (ATPG) is the first tool of its kind which is able to determine test vectors covering resistive shorts for all detectable resistance ranges. RBF-ATPG combines the sectioning technique with an ATPG procedure which exploits a solver for Boolean satisfiability problems (SAT). Based on the data provided by RBF-ATPG, it is now possible to compute exact fault coverages for resistive bridging faults. We were also able to formally derive conditions that guarantee detection of a certain class of resistive bridging faults. Accounting for these conditions drastically simplifies test pattern generation for that class of faults.

We have exploited the opportunities provided by SUPERB and RBF-ATPG to perform numerous experiments. We explored the effectivity of low-voltage testing, low-temperature testing, and their combination. Based on our analytical results we derived guidelines for optimal use of each technique. Furthermore, we contrasted voltage and quiescent

current testing exposing the strengths and weaknesses of each technique. Additionally, we scrutinized the defect detection capabilities of two built-in self test (BIST) techniques. Our procedure may serve as a prototype for a methodology which optimizes the defect detection of BIST architectures. Finally, we investigated the occurrence of double errors induced by resistive bridging faults. Our results have significant influence on the analysis of fault-tolerance techniques.

1.2 Structure of the Thesis

After a general introduction to the field of testing in Chapter 2, the main part of this thesis covers the particularities of bridging faults. The main part is structured into two units. These parts do not differ in their principal topic, rather they both deal with bridging faults. Yet, they approach the topic from two different perspectives.

In Part I we focus on the defect-oriented modeling of resistive shorts. First, in Chapter 3, we give a basic introduction to the terminology commonly used to qualify shorts and their impact on circuit behavior. There, we will also lay the groundwork which allows us to evaluate the bridging fault models to be discussed afterwards. Furthermore, we highlight techniques that identify areas of the circuit which are potentially vulnerable to shorts. Subsequently, Chapter 4 reviews the non-resistive bridging fault models. These conventional models do not explicitly account for the intrinsic resistance of a short. Instead, some of them assume this resistance to be negligible even though it has been proven that a substantial number of shorts has non-zero resistance. Other models try to implicitly represent a short's behavior for any possible intrinsic resistance. We cover several bridging fault models proposed in literature. In the course of our discussion, it will turn out, that neglecting a short's resistance substantially compromises modeling accuracy. Therefore, in Chapter 5 we will focus on the resistive bridging fault model. This parametric model explicitly regards the whole continuum of the intrinsic short resistance. We will demonstrate that by parametrizing the resistance, the modeling accuracy is improved considerably. At the same time, however, the model's revolutionary parametric approach imposes fundamental changes on the meaning of testing concepts such as fault coverage and redundancy. Hence, we will explain how Renovell et al. fitted these concepts to the requirements of the resistive bridging fault model. Likewise, the handling of resistance dependent fault effects is nontrivial. Currently, there are two major approaches to tackle this problem: the interval-based and the sectioning technique. In particular, the latter technique is of great importance for the second part of this thesis.

The core of the resistive bridging fault model is founded by a sophisticated electrical framework. We adapted this framework to two new circuit technology nodes. This adaption is detailed in Chapter 5 as well. There, we also report on our findings on the number of double errors induced by resistive bridging faults. Furthermore, we formulate necessary conditions for guaranteed detection of a certain class of resistive shorts. Finally, Chapter 6 summarizes the insights we gained in Part I.

In Part II we bring testing of resistive shorts into focus. In this part, the efficient algorithms, which enable the integration of the resistive bridging fault model into serviceable tools, are of particular importance. Yet, we also discuss in which way several defect-oriented testing techniques may be exploited for resistive shorts. This is the part of this thesis in which most of our contributions are covered. In Chapter 7 we introduce our high-performance resistive bridging fault simulator SUPERB which renders accurate simulation of multi-million gate designs possible. There, we will detail the algorithms which efficiently combine the sectioning technique with bit-parallel simulation techniques. Chapter 8 introduces RBF-ATPG, our test pattern generator for resistive bridging faults. This chapter thoroughly describes the techniques we have developed to allow the computation of the detectable resistance range of a resistive bridging fault. In the succeeding Chapter 9, we will demonstrate how the electrical framework of the model can be extended to evaluate low-voltage, low-temperature, and quiescent current testing. There, we will also define several fault coverage metrics which enable us to exactly quantify the benefits of these techniques. Subsequently, in Chapter 10, we will explore whether built-in self test techniques impact the coverage of non-target defects. We employ resistive bridging faults as a surrogate for these defects. Finally, Chapter 11 summarizes our contributions presented in Part II.

The Chapter “Concluding Remarks” completes this thesis and sketches several open questions. A compilation of our publications can be found in the Chapter “Author’s Publications” located in the back of this thesis.

2 Preliminaries

This chapter is meant to serve as a brief introduction to the fundamental concepts of digital circuit testing. These concepts are a prerequisite to understand the main part of this thesis. We decided to separate this introduction from the main part as the separation enables the reader to consult a dedicated chapter to look up repeatedly used terms. Advanced concepts and techniques, however, which are only referenced in some chapters, are introduced at first use. Hence, a reader who is familiar with the basics of testing may wish to directly continue reading Part I.

Testing of digital circuits is a well understood discipline with a mature theoretical background. Elaborating every aspect of this field would be far beyond the scope of this thesis. Similarly, it would be cumbersome to detail the theoretical background as this would add complexity, rather than insight. Therefore, we will give an intuitive but concise introduction to testing. Readers who are interested in learning more about the field are directed to one of the many good text books on testing, e.g. [1, 21, 77]. In the following, we first introduce digital circuits in Chapter 2.1. There we will highlight the basic terms and describe the construction principles of digital logic circuits. Then we will switch to testing of digital circuits in Chapter 2.2. After a general introduction this chapter will review the most relevant concepts.

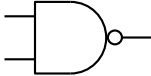

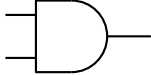

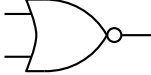
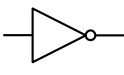

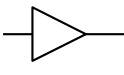
2.1 Digital Circuits

As the principal topic of this thesis is the testing of digital circuits, it is critical that we provide our understanding of this term. Chapter 2.1.1 informally introduces combinational and sequential digital circuits. The chapter also establishes the basic concepts used to describe the components of digital circuits. We will focus on circuits constructed in the complementary metal-oxide-semiconductor (CMOS) technology. The basic principles of this technology are reviewed in Chapter 2.1.2 (more information can be found in e.g. [10, 201]). As the structure of digital circuits is very complex, several levels of abstraction are used to reduce complexity – they are summarized in Chapter 2.1.3.

2.1.1 Combinational and Sequential Circuits

In the context of this thesis we will regard digital *circuits* as devices which process *Boolean* logical values, i.e. values from the set $\mathbb{B} = \{0, 1\}$, sometimes also referred to as bits. A

Table 2.1: Symbols of different types of logic gates.

Symbol	Name	Symbol	Name
	NAND		XNOR
	AND		XOR
	NOR		NOT
	OR		BUF

combinational circuit expects a set of logical values at its *primary inputs* (PI). The circuit maps these values to a set of logical values which can be observed at its *primary outputs* (PO). Let the combinational circuit C have n primary inputs and m primary outputs. The Boolean function computed by C is defined as $f_C : \mathbb{B}^n \rightarrow \mathbb{B}^m$. Digital circuits are implemented using one or more *logic gates*, each of which calculates a Boolean function. The function computed by a gate is dependent on its *type*. A *gate library* is a collection of gate types. In general, the gate library is provided by the manufacturer of the circuit (e.g. a foundry). When constructing a circuit, we are only allowed to use those gate types which are contained in the respective gate library.

In the following, we will assume a small gate library that contains the standard gate types, which in general we can expect to be part of every larger industrial gate library. Table 2.1 lists some of the gate types which we will consider. Columns labeled “Symbol” denote the graphical representation used in all figures; columns labeled “Name” state the name of the respective gate type. For all gates listed in the two leftmost columns of the table, i.e. NAND, AND, NOR, and OR, we will also allow instances with more than two inputs. Occasionally we will refer to the NOT gate as to an *inverter*. Furthermore, our gate library contains a primary input and a primary output which represent the connection of a circuit to its environment. A primary input has no inputs and one output and feeds a logical value supplied by the environment into the circuit. Hence, primary input i , where $1 \leq i \leq n$, provides the i -th component $x_i \in \mathbb{B}$ of the bit-string (x_1, \dots, x_n) to be processed by the circuit. Inversely, a primary output exhibits only one input and no output. A primary output makes a logical value computed by the circuit accessible to the environment. Primary output j , where $1 \leq j \leq m$, makes the j -th component z_j of the bit-string (z_1, \dots, z_m) observable to the environment. We say output (z_1, \dots, z_m) has been computed by the circuit in response to the bit-string (x_1, \dots, x_n) assigned to its primary inputs.

Figure 2.1(a) depicts a combinational circuit C with three primary inputs in_1 , in_2 , and in_3 and two primary outputs out_1 , and out_2 . Primary inputs and outputs are indicated by a small square symbol. Furthermore, the circuit consists of three logic gates A , B and C . In combination, the logic gates compute the following function (where \neg , \vee , and \wedge represent

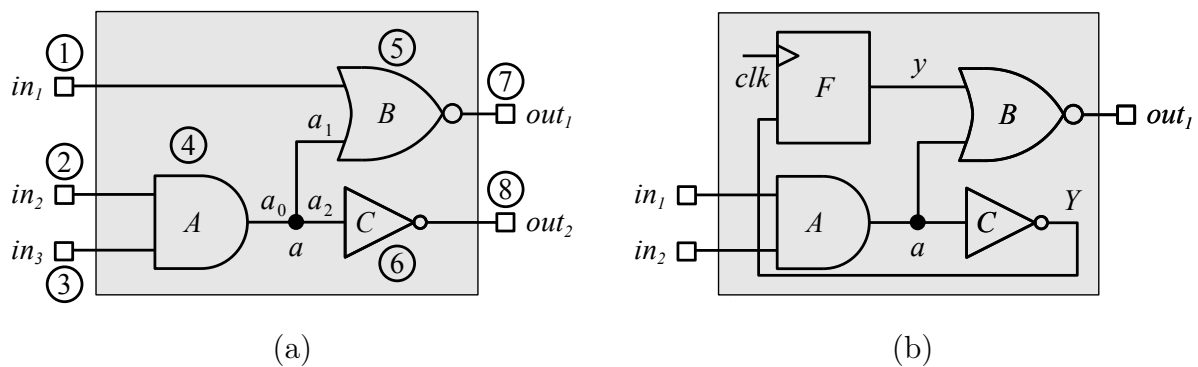


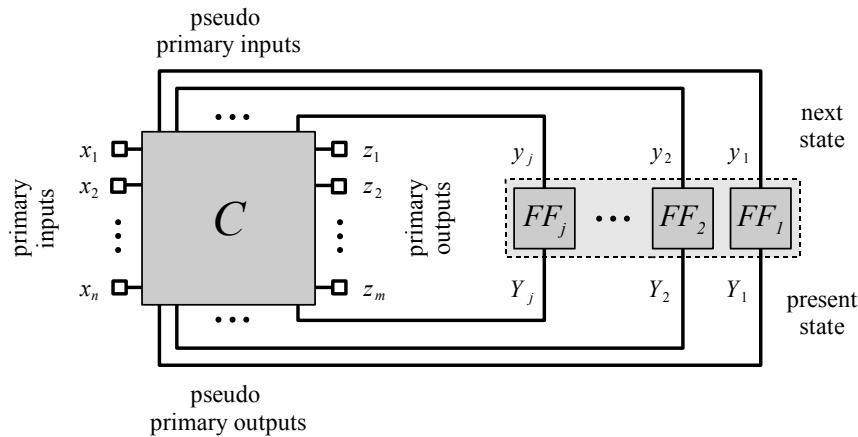
Figure 2.1: Example of (a) combinational and (b) sequential circuit.

NOT, OR, and AND operations, respectively): $f_C = (\neg(in_1 \vee (in_2 \wedge in_3)), \neg(in_2 \wedge in_3))$. We associate a *circuit node* with the output of every logic gate (this includes the primary inputs). Each circuit node is connected to exactly one gate output – we say the node is driven by that output – and to one or more gate inputs. The connection between a gate’s output and a circuit node is referred to as *stem*. Each connection between a circuit node and a gate input is called a *fanout branch*. In Figure 2.1(a) the stem of node a is indicated as a_0 , its two fanout branches are marked by a_1 and a_2 , respectively.

If we assign the bit-string $(0, 0, 1)$ to the primary inputs in_1 , in_2 , and in_3 of our example circuit, the logical values $(1, 1)$ computed according to function f_C will be observable at its primary outputs out_1 and out_2 . To obtain this result we have to *propagate* the logical assignments through the circuit visiting the gates in *topological order*. Propagation for a gate g means that we apply the Boolean function associated with g to the logical values observed at the nodes which are connected to the inputs of g . Subsequently, we assign the resulting logical value to the node driven by the output of g . The topological order t_C of a circuit C is a mapping which enumerates all gates G contained in C , i.e. $t_C : G \rightarrow \mathbb{N}$. Function t_C for a gate $g \in G$ is defined such that for all gates $g' \in G$, that are driving nodes directly connected to the inputs of g , it holds that $t_C(g') < t_C(g)$. Gates without inputs are assigned the lowest topological number that is so far unallocated. A possible topological order for the gates in our example circuit from Figure 2.1(a) is indicated by the numbers in circles.

We will use the term *path* to denote a well-defined sequence of logic gates which are connected via nodes. A path starts at the output of a logic gate and ends at a primary output. In the circuit from 2.1(a), primary input in_2 , gates A and B , and the primary output out_1 are on a path. A *fanout cone* associated with a gate g subsumes all gates lying on any arbitrary path which starts at g and ends at some primary output. The fanout cone of gate A in the circuit from 2.1(a) encompasses gates B and C and primary outputs out_1 and out_2 .

A *sequential circuit* is a circuit which contains memory elements, i.e. so-called flip-flops. Figure 2.1(b) illustrates such a circuit with flip-flop F . We assume that this special gate type is also contained in our gate library. Just like combinational circuits the sequential ones process Boolean values provided at their primary inputs. The output of sequential

Figure 2.2: Model of a synchronous sequential circuit S .

circuits is, however, not only dependent on those input values but also on the circuit's internal state, i.e. the logic values stored in the flip-flops. The sequential circuit from Figure 2.1(b) is very similar to the one depicted in 2.1(a). Yet, in contrast to the latter combinational circuit, the logical value computed by gate C is not observable at a primary output. Rather, the value is stored in flip-flop F which is connected to C via node Y . Gate F drives node y and thus feeds the stored logical value back into the circuit to the upper input of gate B . The flip-flop only stores a value if the *clock signal* clk is active otherwise it will provide the retained logical value.

To compute the response of our sequential circuit to an assignment to its primary inputs, we have to know the value stored in flip-flop F . For the moment we will assume that F retains logical value 0 and that $(0, 1)$ is assigned to primary inputs in_1 and in_2 . Then, a logical value 1 will be observable at the circuit's primary output out_1 . Furthermore, logical value 1 is driven at the input of flip-flop F . If we now activate clock signal clk , gate F will replace its former contents by the new logical value 1. Provided that we keep the input assignment $(0, 1)$ constant, logical value 0 will now be observable at primary output out_1 . If we neither can make assumptions about the contents of a flip-flop nor are able to control its contents (e.g. by performing a reset operation), we say that the flip-flop stores an *unknown value*. The unknown value, often denoted by symbol "X", can represent either logical value 0 or 1. If an unknown value is stored in flip-flop F of our example circuit and we continue to assign $(0, 1)$ to its primary inputs, the value observable at the primary output will be unknown as well.

In the following, we will restrict the class of sequential circuits to *synchronous sequential circuits*. These are circuits in which all flip-flops are controlled by a clock signal. To model a synchronous sequential circuit, we separate the flip-flops from the combinational (core) logic. Now, we assume that the flip-flops are replaced by ideal memory elements. Controlled by the clock signal, these elements either store the logical values provided by the combinational core or feed their contents back into the core. This concept is illustrated in Figure 2.2 for circuit S with n primary inputs, m primary outputs, and j flip-flops FF_1, FF_2, \dots, FF_j . The outputs y_1, y_2, \dots, y_j of the flip-flops serve as *pseudo primary*

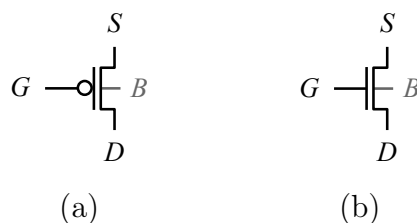


Figure 2.3: Symbol of (a) p -channel and (b) n -channel MOS-FET.

inputs (PPI) to the combinational core C of S . They provide the current content of the memory elements which is processed by the combinational core C . When the clock is active, the values present at flip-flop inputs Y_1, Y_2, \dots, Y_j are stored in the memory elements. These inputs are treated as *pseudo primary outputs* (PPO) of the combinational core, which take the “present state” of the circuit. Subsequently, the new content of the memory elements, i.e. their “next state”, is fed back into the circuit. The function computed by the combinational core C of circuit S with j flip-flops can be written as $f_S : \mathbb{B}^{n+j} \rightarrow \mathbb{B}^{m+j}$.

Up to now, we have only discussed how circuits operate in the domain of Boolean values. Yet, there is also a time aspect to their operation. Each gate requires a certain amount of time, called *propagation delay*, for the gate’s output to respond to a change of the logical values assigned to the gate’s inputs. In general, the propagation delay is specified as a range rather than a fixed value and is given individually for each gate type. The *signal propagation delay* is the time elapsing until the primary outputs of a circuit have stabilized to correct logical values in response to a change of the logical values assigned at the circuit’s primary inputs. The signal propagation delay must be larger than the maximal total propagation delay on any path from a (pseudo) primary input to a (pseudo) primary output. In sequential circuits, the signal propagation delay determines the *clock cycle*, i.e. the phase in which the clock signal toggles once between active and inactive state. This in turn dictates the frequency at which new logical values are applied to the inputs of the circuit.

2.1.2 Implementation of Digital Circuits

In this thesis we will focus on the *complementary metal-oxide-semiconductor* (CMOS) [200] *logic family*. This chapter is meant as a brief introduction to the principles of this logic family, details can be found in e.g. [10, 201]. We will use the term logic family to refer to a certain technology used to implement digital circuits. This does not necessarily presume interoperability or electrical compatibility of devices implemented in that technology. The building-block of the CMOS logic family is the logic gate. Each logic gate consists of a number of transistors. These transistors are *metal-oxide-semiconductor field-effect transistors* (MOS-FET) which come in two different versions: The *p-channel* and the *n-channel* MOS-FET. Most of the time we will refer to them as *p-transistor* and *n-transistor*, respectively. MOS-FETs have four terminals called *gate* (G), *source* (S), *drain* (D), and *bulk* (B). In many applications bulk and source terminal are held at the same voltage

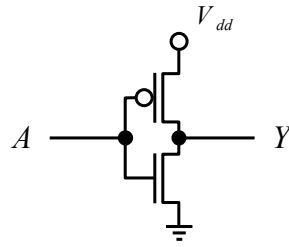


Figure 2.4: Standard CMOS implementation of a NOT gate with input A and output Y .

potential. Contrary to that, source and bulk may attain different potentials in CMOS. Commonly *positive supply voltage* V_{dd} is applied to the bulk terminal of a p -transistor and *negative supply voltage* V_{ss} (often 0 V, i.e. ground) is supplied to the bulk terminal of an n -transistor. Throughout this work we use the simplified symbols depicted in Figure 2.3(a) and 2.3(b) to denote p - and n -channel MOS-FET, respectively. We will omit the B terminal in these symbols.

A voltage applied to the gate G (more precisely a difference in the voltage potential between G and B terminal) controls the current flowing through the path from drain to source – provided that there is a difference in the voltage potential between the D and the S terminal. The amount of current conducted is correlated to the difference of the voltage potential between G and B . The p - and the n -transistor have a complementary relation between voltage and current. The p -transistor allows current to flow whenever a low voltage is applied to its G terminal, i.e. there is a sufficiently large difference in the voltage potential between G and V_{dd} (applied to the B terminal). In this case, we say the device is *activated*. If the voltage difference between these terminals is sufficiently close to zero the current path is blocked – we say the device is *deactivated*. Contrary to that, the n -transistor is activated whenever a high voltage is present at its G terminal. This means, that the difference of the voltage potential between G and ground potential at B is sufficiently large.

Each input of a (basic) logic gate (such as AND, NAND, OR, NOR, NOT, and BUF) of the CMOS logic family is connected to the gate terminal of at least one p - and one n -channel MOS-FET. Utilization of a (symmetric) set of complementary MOS-FETs led to the name “complementary metal-oxide-semiconductor” for this technology. As a consequence the number of transistors in a (basic) logic gate with i inputs is $2 \cdot i$. Figure 2.4 depicts the standard CMOS implementation of an inverter, i.e. a NOT gate, with input A and output Y . The p -transistor connects the power supply terminal V_{dd} to the inverter’s output Y . The n -transistor links the ground terminal – indicated by a triangular symbol (sometimes also denoted by GND or V_{ss}) – to output Y . The G terminals of both transistors are connected to input A of the inverter. When A is set to low voltage potential (sufficiently close to ground potential), which is identified with logical value 0, the p -transistor is activated. Thus, it establishes a connection between V_{dd} and Y . At the same time, the n -transistor is blocked. Current supplied through the V_{dd} terminal can now be distributed to all G terminals of all transistors in the logic gates connected to Y . These transistors can be seen as a capacitive load which is charged to an elevated voltage potential (close to V_{dd}).

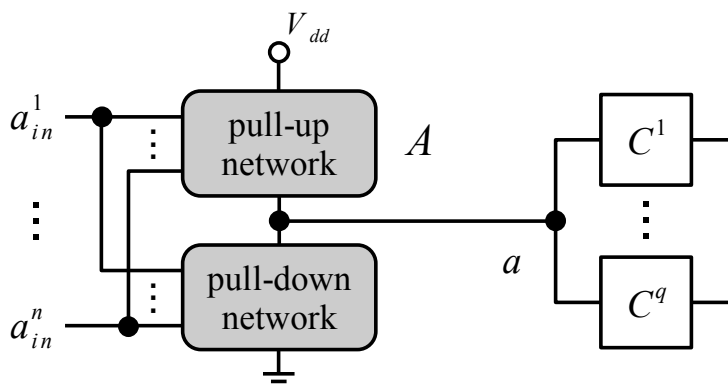


Figure 2.5: Pull-up and pull-down network of a logic gate.

This potential is identified with the logical value 1. Consequently the n -transistors in the logic gates connected to Y are activated – we say the gates interpret the logical value 1. If on the other hand input A of the inverter is set to an elevated voltage potential, the n -transistor is activated while the p -transistor is blocked. As a consequence a current path between output Y and the ground terminal is established. It discharges the capacitive load formed by all transistors in the logic gates connected to Y , thus creating a low voltage potential (close to ground potential). Hence, the p -transistors in these gates are activated – we say the gates interpret logical value 0. This explanation confirms that the logic gate depicted in Figure 2.4 indeed implements the functionality of an inverter.

A logic gate g interprets the voltage at any of its inputs with respect to the input's *logic threshold voltage*. Let V^o be the voltage driven at output o of logic gate g . In accordance with e.g. [3, 146] we define the logic threshold voltage V_{lt}^i of input i of g as the voltage which has to be applied to i in order to obtain $V^o = V_{\text{lt}}^i$. This definition assumes that all other inputs of the logic gate are held at a stable voltage corresponding to the non-controlling value. Each input of each gate type may have a distinct logic threshold voltage. Gates of different types may have different (sets of) thresholds. There may even be several versions of the same gate type, e.g. several two-input NAND gates, having different sets of logic thresholds. In the absence of noise, any voltage at i which is lower than V_{lt}^i will be interpreted as logical value 0. Any voltage larger than V_{lt}^i will be interpreted as logic 1. Voltages which are very close to the threshold value may not always be clearly resolvable.

According to the design principle of a (basic) CMOS logic gate, there will always at least one, sometimes several interconnected p -transistors forming a so-called *network*. This *p-transistor network* connects the V_{dd} terminal to the output of the logic gate. Furthermore all n -transistors form a network – the *n-transistor network* – which links the output of the logic gate to the ground terminal. As the p -transistor network establishes an elevated voltage potential at the output of the logic gate, it is often called a *pull-up network*. Contrary to that, the n -transistor network enforces a low voltage potential at the output and is thus called a *pull-down network*. Provided that the voltages at the inputs of the logic gate are stable and sufficiently close to either low or high voltage potential, the output of a defect-free CMOS logic gate is always at either low or high voltage potential. Figure 2.5

depicts pull-up and pull-down networks of the n input gate A driving q gates C^1 to C^q .

Even though the general construction principle of CMOS logic gates has remained virtually unchanged, the manufacturing technology is always evolving. The minimum size of a feature, which can be used to construct the transistors of a logic gate, is defined by the *technology node*. Currently, circuits implemented in 45 nm technology are in mass production. But the potential for downscaling the feature sizes is not exhausted yet and future technology nodes are already under development.

2.1.3 Levels of Abstraction

Digital circuits are very complex devices which are composed of a gigantic number of structures. Circuit behavior, which is a result of the interaction of all these structures, is governed by intricate physical phenomena. Thus, it is impossible to regard a circuit containing several million gates in every detail of its implementation. However, in many steps of the design process of such a circuit, this richness of detail is not necessary, and only entails excessive complexity. Therefore, at each design step the circuit may be considered at those *levels of abstraction* matching the needs of that step. The idea of abstraction is to create a simplified model closely representing the original behavior of a circuit. This means that certain implementation details are hidden which reduces complexity of the circuit description.

In literature there is no definite set of levels of abstraction. Rather, several authors adapt the concept to their needs (see [50, 67, 196] for some of the approaches). Since we focus on the testing part of the circuit design process, a selection of abstraction levels is sufficient. These levels are – in descending order – gate-level, switch-level, transistor-level, and layout-level. They are defined as follows:

Gate-level: The gate-level representation of a circuit has already been covered in Chapter 2.1.1. At this level of abstraction, a circuit is regarded as a collection of connected logic gates, flip-flops, and nodes. This collection forms the so-called gate-level netlist. At the gate-level the circuit processes Boolean values provided at its primary inputs. Logical values computed according to the circuit's Boolean function are observable at its primary outputs.

Switch-level: At the switch-level, the representation of logic gates and flip-flops is refined. We now consider the network of transistors forming a logic gate instead of assuming the gate to be a monolithic block. Each transistor is controlled by voltages which correspond to the logical values employed at the gate-level. A gate's transistors are modeled as ideal switches. These switches are operated by voltages which are applied to the inputs of a gate. A network of activated switches couples the output of a gate to either positive or negative supply voltage. All transistors in a circuit are connected by wires, and electrical contacts link the wires to the circuit's environment.

Transistor-level: In contrast to the switch-level, the transistor-level models transistor behavior in more detail. A transistor is no longer an ideal switch. Rather, its

behavior depends on the voltages and currents which are applied to its terminals. Furthermore, resistors and capacitors may be connected to the terminals of transistors. The description of a circuit at this level is referred to as transistor-level netlist. Each component of the netlist may be represented by differential equations. They try to replicate a circuit's electrical behavior. By performing an *electrical simulation*, we can determine the electrical behavior of a circuit. This process essentially involves solving the differential equations describing the circuit's components. Inputs to this process are the currents and the voltages applied to the electrical contacts.

Layout-level: The actual integrated digital circuit consists of several conducting, non-conducting, and semi-conducting layers. During production, the structures composing each layer are transferred to a piece of silicon using photolithographic processes. These processes employ masks on which geometric shapes prevent radiation from reaching the silicon. The geometric shapes on each mask are represented by the so-called *layout*. At the layout-level wires and transistors of a chip are described as a collection of polygons. We refer to the layout representation of a wire as an interconnect.

In general, an algorithm used in testing of digital circuits can be more efficient if it operates on a high level of abstraction. Therefore, nearly all of the techniques covered in this thesis operate on the gate-level. However, to be able to accurately represent defects, more information is required than can be derived from the gate-level. Thus, to increase accuracy we will occasionally have to resort to lower abstraction levels. For instance, in Chapter 3.3, we will identify circuit nodes which have a high probability of being affected by a short at the layout-level. Furthermore, in Chapter 5, we will determine the consequences of a resistive short on circuit behavior by considering the gates affected by that defect at the transistor-level.

2.2 Testing of Digital Circuits

In the next chapters we will discuss the fundamental concepts of circuit testing in more detail. First Chapter 2.2.1 will summarize the general terminology used in the main part of this thesis. The particularities of testing sequential circuits are highlighted in Chapter 2.2.2. The contents of this chapter enable the reader to understand the discussion of our results on Delta-IDDQ testing elaborated in Chapter 9.2. Then we define the "traditional" stuck-at fault model in Chapter 2.2.3. Moreover, we will also formulate two conceptual extensions to this model: namely multiple-stuck-at and conditional multiple-stuck-at faults. These formalisms are exploited extensively in the chapters to come. Finally, in Chapter 2.2.4 we lay the groundwork for the built-in self test technique which may simplify testing of circuits considerably. Our experiments in Chapter 10 verify if this advantage is paid for by reduced defect detection.

2.2.1 General Terminology

In the following we will understand a *fault* as a representative of a physical defect. In general a defect may result in a deviation of the observed circuit behavior from the expected behavior. A *fault model* reflects the impact of a physical defect on the behavior of a circuit, we say it models the *fault effect*. We will consider *structural* fault models. These fault models act on the assumption that the structure of the circuit is known to us. They also assume that a fault may only affect the connections between gates while the logic function of each individual gate is intact. Furthermore, we will focus on verifying the Boolean behavior of a circuit. This is referred to as *static testing* or static voltage testing, as during test application voltage levels representing Boolean values are measured. Fault models, which target circuit behavior in the time domain, are not covered in this thesis.

In general, similar defects may affect different parts, e.g. nodes, of the circuit. Each of these defects will be represented by a different fault. We call a defect-free circuit a *good circuit*. The circuit, however, which contains a defect is labeled the *defective circuit* or *faulty circuit*. In a defective circuit, a defect may deviate the logical value observed at some circuit node n from the value seen in the good circuit at the same node. We say the logical value present at node n in the latter circuit is a *fault-free logical value* (or fault-free value) whereas the logical value seen at n in the faulty circuit is referred to as the *faulty logical value* (or faulty value). A node, which displays a faulty value, is denoted as *faulty node*.

Let C be a combinational circuit with n primary inputs and m primary outputs which implements function $f : \mathbb{B}^n \rightarrow \mathbb{B}^m$ and let l be a fault. We denote an instance of C which contains the defect represented by l as C_l . Due to the defect, the function $f_l : \mathbb{B}^n \rightarrow \mathbb{B}^m$ implemented by C_l may be different from the intended function f . A *test vector*, *test pattern*, or simply *test* is a bit-string $x \in \mathbb{B}^n$ such that $f(x) \neq f_l(x)$. We say that x *detects* or *covers* fault l . If there is no test pattern for fault l , i.e. for all 2^n possible bit-strings $x \in \mathbb{B}^n$ we obtain $f(x) = f_l(x)$, we say that l is *redundant*. The process in which, given a fault l , we try to determine a test x such that $f(x) \neq f_l(x)$ is referred to as (*automatic*) *test pattern generation* (ATPG). Inversely, we may already have a bit-string x and a set of faults F and wish to determine the set $F^{\text{det}} \subseteq F$ of faults which are detected by x . This process is called *fault simulation*. In this context, we are often interested in a single figure which quantifies the percentage of faults from F covered by x . This figure is provided by the *fault coverage* (FC) defined as follows ($|X|$ denotes the number of elements contained in set X):

$$\text{FC} = 100\% \cdot \frac{|F^{\text{det}}|}{|F|} \quad (2.2.1)$$

To compute the fault coverage of a set of s test vectors $T = \{t_1, t_2, \dots, t_s\}$ we first derive the set of faults $F^i \subseteq F$, $1 \leq i \leq s$, detected by every individual vector t_i . Subsequently, we join the sets F^i to obtain the complete set F^{det} of detected vectors, i.e. $F^{\text{det}} = \bigcup_{i=1}^s F^i$, and compute the fault coverage according to Equation (2.2.1).

The expressiveness of the fault coverage can be refined if we know the set F^{red} of redundant faults contained in F (note that $F^{\text{red}} \subseteq F$). The *fault efficacy* (FE), sometimes also referred

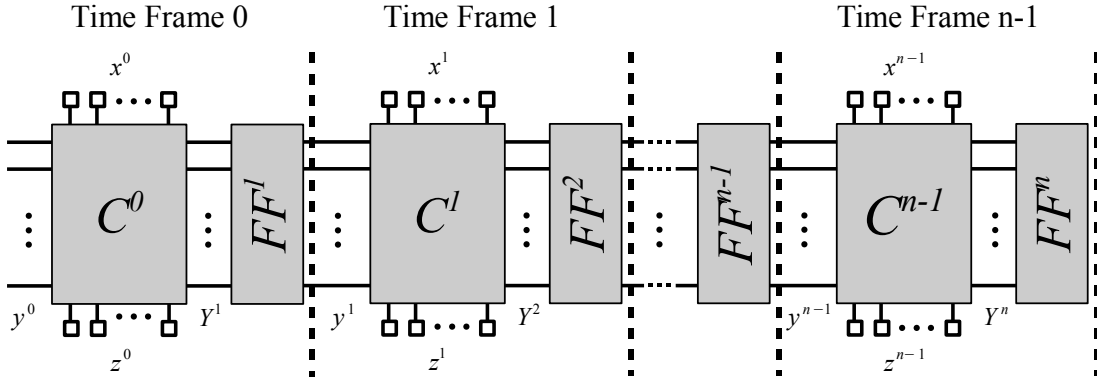


Figure 2.6: Time-frame expansion of a synchronous sequential circuit.

to as *fault efficiency*, quantifies the percentage of detectable faults F^{det} which are covered by test vector x (if $F^{\text{red}} = F$ the fault efficacy evaluates to 0%):

$$\text{FE} = 100\% \cdot \frac{|F^{\text{det}}|}{|F| - |F^{\text{red}}|} \quad (2.2.2)$$

As we have to know F^{red} in order to be able to compute the fault efficacy this metric is not as universal as the fault coverage. In particular, it is very problematic to identify the set of redundant faults for sequential circuits.

2.2.2 Testing of Sequential Circuits

The difficulty of testing sequential circuits arises from the flip-flops. Just as for combinational circuits, to test for a fault we have to place specific values at the circuit's primary inputs. In addition to that, however, to test a sequential circuit we have to control the values present in the flip-flops as well. Yet, we can only manipulate the contents of the flip-flops from the primary inputs. Furthermore, even if we manage to activate a fault, it might be impossible to directly propagate any of its effects to a primary output. Rather, the fault effects may have to pass through one or more flip-flops before we can actually observe them. As a consequence, in general we have to test faults in sequential circuits by a sequence of primary input assignments. This complicates ATPG substantially and may even prevent test pattern generation for certain faults. Therefore, unlike for combinational circuits, a fault in a sequential circuit is not necessarily redundant if it is untestable.

The *time frame expansion* is one method to facilitate test pattern generation and fault simulation for sequential circuits. This method uses a combinational iterative array concept which exploits the model of a synchronous sequential circuit depicted in Figure 2.2. The sequential circuit is again split into two parts, i.e. the memory elements and the combinational core. An n frame time frame expansion is depicted in Figure 2.6. Within *time frame* i , where $0 \leq i \leq n - 1$, combinational core C^i processes the primary input assignment x^i and the next state y^i which has been stored in the preceding time frame. At the end of the clock cycle, response z^i is observable at the primary outputs of C^i . Moreover

the circuit's present state Y^{i+1} is written to the memory elements FF^{i+1} . Hence, one time frame models the state of a sequential circuit for one clock period. In each time frame one test pattern is applied to the primary inputs and one circuit response may be observed at the primary outputs. Thus, for one time frame, test pattern generation and fault simulation are very similar to their equivalents for combinational circuits.

The time frame expansion is a software concept which simplifies the algorithmic part of test generation. Yet, it does not ease the test generation itself. This means that it is still difficult to control the contents of the flip-flops and propagate the fault effects. This problem is drastically reduced by so-called *scan chains*. A scan chain connects (some of) the flip-flops contained in a circuit in a serial manner, i.e. it is a solution implemented in hardware. The start and the end of a scan chain are connected to a primary input and a primary output, respectively. This provides the ability to serially shift values provided at this primary input into the flip-flops organized in the chain. At the same time, the contents of these flip-flops are serially shifted out to the primary output. Hence, we can observe the values stored in the flip-flops connected to a chain. Therefore, a scan chain allows us to control and observe the contents of all flip-flops arranged in that chain. As a consequence, we may treat the pseudo primary inputs and outputs corresponding to these flip-flops just like regular primary inputs and outputs. In the following we will assume all circuits to be *full scan circuits*. This means that every flip-flop in these circuits is part of a scan chain. From the perspective of test pattern generation and fault simulation there is no difference between a full scan circuit and a combinational circuit. Whenever we do not explicitly want to distinguish between primary outputs and pseudo primary outputs we will subsume both as *observable points*. In summary, a scan chain is a hardware concept which is specifically designed into a sequential circuit to enhance its testability. Hence, scan chains are often called a *design for testability* (DFT) measure.

2.2.3 Stuck-At Faults

The most common fault model used both in industry and academia is the *stuck-at fault model*. The stuck-at fault model assumes that due to some defect, circuit node v permanently assumes a fixed logic value $v^{\text{val}} \in \mathbb{B}$ – for any logical assignment to the primary inputs of the circuit. We say that v is stuck-at v^{val} or in short “ v s-a- v^{val} ”. A test x which detects v s-a- v^{val} must induce value $\overline{v^{\text{val}}}$ at node v ; we say x *activates* the fault. Furthermore x must propagate the fault effect v^{val} such that a faulty value is observable at one or more observable points of the circuit, e.g. primary outputs.

Typically it is assumed that only a single (stuck-at) fault, i.e. a single relevant defect, is present in a circuit at any time – we will retain this assumption. Yet, as we want to use the concept of stuck-at faults to represent more complex fault models introduced in Part I of this thesis, we allow stuck-at faults affecting multiple circuit nodes. A *multiple-stuck-at* (MS@) fault s affecting $m \in \mathbb{N}_{>0}$ nodes v_1, v_2, \dots, v_m permanently induces value $v_i^{\text{val}} \in \mathbb{B}$ at v_i , where $1 \leq i \leq m$. We specify multiple-stuck-at fault s as $\{v_1/v_1^{\text{val}}, v_2/v_2^{\text{val}}, \dots, v_m/v_m^{\text{val}}\}$. A test x which detects s must activate at least one fault v_i s-a- v_i^{val} and make a faulty value observable at one or more observable points of the circuit. Whenever appropriate

we will refer to the common stuck-at fault model which assumes a single fault site as to the *single-stuck-at fault* model. Note that for $m = 1$ a multiple-stuck-at fault actually represents a single-stuck-at fault.

In some cases, fault effects may only be induced by a defect if specific logical values are present at a set of circuit nodes. To represent such a situation we can use a *conditional multiple-stuck-at fault* (CMS@). Basically CMS@ faults are an extension of multiple-stuck-at faults. A CMS@ fault s combines a multiple-stuck-at fault consisting of $m \in \mathbb{N}_{>0}$ stuck-at faults or *victims* $\{v_1/v_1^{\text{val}}, v_2/v_2^{\text{val}}, \dots, v_m/v_m^{\text{val}}\}$ with $l \in \mathbb{N}$ so-called aggressors $\{a_1/a_1^{\text{val}}, a_2/a_2^{\text{val}}, \dots, a_l/a_l^{\text{val}}\}$. Each aggressor is defined by a node a_i and a value $a_i^{\text{val}} \in \mathbb{B}$, where $1 \leq i \leq l$. In order to detect fault s , test x must assign to each aggressor a_i the respective value a_i^{val} . Furthermore it must detect the multiple-stuck-at fault $\{v_1/v_1^{\text{val}}, v_2/v_2^{\text{val}}, \dots, v_m/v_m^{\text{val}}\}$ as described above. Note that a CMS@ fault with empty aggressor list, i.e. $l = 0$, can be represented by a multiple-stuck-at fault.

2.2.4 Built-In Self Test

Traditionally, circuits are tested by *automatic test equipment* (ATE). The ATE applies a set of test patterns which is stored in its internal memory to the *circuit under test* (CUT). Furthermore, it samples the response of the CUT to each pattern and compares the sampled values to the response expected for that pattern. As soon as the observed response does not match the reference values, the CUT is said to have failed the test and is considered defective. Since test application and response evaluation are performed by the ATE, the speed of this device determines the frequency at which the CUT may be tested. This, however, means that in order to test high-performance parts at their dedicated clock frequency, the ATE has to be faster than these parts. ATEs which can keep pace with high-speed circuits are extremely costly and may even be unavailable for the fastest circuits in production.

Built-in self test (BIST) is a DFT measure which transfers test pattern application and response evaluation onto the chip. This drastically reduces the performance requirements for the ATE, and in particular, enables at-speed test of high speed circuits which may facilitate the detection of certain defects. BIST circuits can also perform autonomous self-test in the field, i.e. once they have been integrated into a system. This is crucial for high reliability systems.

A basic BIST architecture is depicted in Figure 2.7. It consists of a *pattern generator* (PG) which feeds test patterns to the CUT. Circuit responses are evaluated by the *test response evaluator* (TRE). Both PG and TRE are operated by the *BIST control* block. The control block is linked to the external ATE which initiates and monitors the self test procedure. This communication channel may be omitted if the circuit is performing an autonomous self-test. Further communication channels between PG and ATE, and TRE and ATE, respectively, may be required depending on the BIST architecture. It is obvious that BIST adds hardware to the CUT which increases silicon area demands and thus production cost. Beyond that, parts of the BIST hardware may introduce additional delay on certain

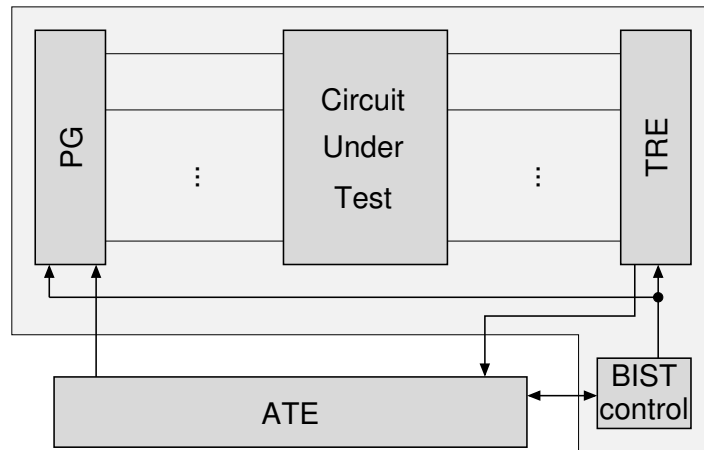


Figure 2.7: Basic built-in self test architecture.

functional paths in the circuit. This in turn reduces the maximum operating frequency of the device.

For both the PG and the TRE several different designs have been proposed in literature. We will only highlight one particularly prominent instance of each component. Pattern generators are often implemented as *linear feedback shift registers* (LFSR). An LFSR consists of a shift register with a feedback connection implemented using XOR gates. The shift register of the LFSR runs through a repeatable sequence of states. The contents of the shift registers, i.e. the state variables, can be supplied as test vectors to the CUT. As the sequence generated by an LFSR fulfills many empirical properties for randomness it is often referred to as *pseudo random* test pattern source. LFSRs can be designed to comply with a variety of criteria. Often it is desirable to achieve completeness which means that all 2^n test patterns possible for a CUT with n inputs can actually be generated by the PG. Furthermore, it may be required that each pattern is contained at most once in the generated sequence, this is referred to as uniqueness. In some applications only a part of the LFSR generated sequence is applied to the CUT. Hence, it is beneficial if those test patterns are sampled uniformly from the domain of 2^n potentially possible test patterns.

A very popular TRE is the *multiple-input shift register* (MISR) which is virtually an extended LFSR. Just as most of the other TRE designs, the MISR compresses the responses of the CUT on-chip into a so called *signature*. Compression is absolutely necessary as an on-chip comparison with the complete response of the fault-free circuit would require prohibitive amounts of memory. The signatures, however, are rather small. Hence, it is feasible to provide memory for the signatures of a good circuit on the chip. This enables response evaluation by the BIST hardware. On the other hand, a signature may also be transferred in certain intervals to the ATE and can thus be evaluated externally. In this case the fault-free signatures are stored off-chip such that no memory is required for the TRE. As compression is a lossy process, it is possible that two responses of the CUT are represented by the same signature. This effect can lead to *aliasing*, i.e. the phenomenon that a fault-free and a faulty response are both mapped to the same signature. If possible aliasing should be avoided, as otherwise defective circuits may remain undetected.

Part I

Bridging Fault Models

3 Fundamentals of Bridging Faults

Several studies [45, 51, 66, 169] have reported that a substantial number of defects manifest themselves as shorts between conducting elements of the circuit. This is not a new development but has been known for many years. Fault models for shorts were already mentioned in the 1960s (e.g. in Roth’s fundamental work on ATPG [161]). By that time, it was already obvious that the characteristics and the effects of this defect class cannot be captured satisfyingly by the common stuck-at faults. Therefore a new fault model was needed: The *bridging fault model*. In the following Chapter 3.1 we will subsume the observations that led to this conclusion and casually introduce the basic terminology used in the field of bridging fault testing. Subsequently, in Chapter 3.2 we will motivate the classification of bridging fault models used throughout this thesis. Finally, in Chapter 3.3 we will describe how potential locations for short defects can be derived from the layout of a given circuit.

3.1 Basic Properties and Terminology

In 1973 Williams et al. [202] summarized several characteristics of shorts that are still valid today. They demonstrated that the effects caused by an unwanted connection between nodes a and b , i.e. a short, can only be observed when contrary logical values are driven on the two nodes, i.e. a driven to logical value 1 and b to logical value 0 or vice versa. As the nodes are connected by a conducting defect, only contradicting logical values lead to an electrical conflict (called drive fight or logic contention in [146]) which may expose the short. A test assigning the same logical value to both a and b would not make the defect detectable.

In principle, shorts involving more than two nodes, i.e. $p > 2$ nodes, are possible. Depending on the logic family used to construct the circuit, they can be modeled by a composition of $p - 1$ two-node bridging faults.[1, p. 290] In the following we will adopt the common assumption of two-node shorts.

In [202] it is also reported that a short between the outputs of two gates may affect the electrical state of the shorted nodes such that even gates connected to the same node might not interpret a consistent logical value any more. This would later be termed *Byzantine General’s Problem* [4] or *Byzantine behavior*.

Williams et al. also realized that the number of two-node shorts – and thus the number of bridging faults – theoretically possible in a circuit can be enormous. This was rendered more precisely by Mei in [115]:

“The total number of possible bridging faults in a circuit with n leads (ed.: nodes) is $\binom{n}{2}$.”

Williams et al. advised to focus on the subset of shorts affecting interconnects physically adjacent in the circuit’s layout, as these are the most likely defect locations. We will refer to these defects as *layout extracted shorts* (or *layout extracted bridging faults* when relating to their modeling). In general we will use the term *short* to denote an actual defect. The term *bridging fault* or *bridge* refers to the model representation for one or several shorts which affect the same (logical) circuit nodes. Extraction of such shorts has been realized later on by several authors (see Chapter 3.3).

Another property mentioned in the 1973 paper is that shorts may create feedback-loops. According to Mei [115], a so-called feedback bridging fault modeling such a defect is defined as follows:

“A feedback bridging fault is a bridging fault such that both involved leads (ed.: nodes) lie on the same path in the circuit.”

We will refer to the node with the lower topological order as *back line*. The node with the higher topological order will be denoted as *front line*. Under certain circumstances a feedback loop may provoke oscillating logical values on some nodes in the circuit. In other cases state-holding, i.e. sequential, behavior has been observed (see e.g. [2, 65, 89, 115] for a discussion). In general, detection of shorts causing such loops is difficult and requires special techniques. Therefore a separated consideration of *feedback* and *non-feedback* bridging faults (for which no such path exists) is common.

Just like the authors of [165] we distinguish *inter-gate* bridging faults, i.e. bridges affecting the outputs of (two) gates, and *intra-gate* bridges. (In [118] a “bridging fault taxonomy” with finer granularity is reported.) The latter type of shorts involves conducting elements of the gate’s internal structure only. Our discussion of bridging faults will be restricted to the inter-gate type.¹ Approaches tackling intra-gate bridging faults may be found in [36, 123, 145, 153, 173].

One property of a short not mentioned so far is its *intrinsic resistance*. This resistance is primarily dependent on the defect’s size, location, and material. It has a substantial influence on the impact of the short and thus on the behavior of the defective circuit. A short with low intrinsic resistance will permanently change this behavior and can thus be detected by *static testing* methods. Yet, the larger the intrinsic resistance the less influential the coupling between the two shorted nodes will be. A short with sufficiently large resistance may solely affect the dynamic behavior of circuit and can thus be detected by *delay testing* methods only. We will report on bridging fault models explicitly considering the intrinsic resistance and call them *resistive bridging fault* (RBF) models. We will also discuss *non-resistive bridging fault* (NRBF) models which neglect the impact of the intrinsic

¹This is supported by an experimental study in [177] which provides evidence that the predominant share of shorts affect the outputs of different logic gates.

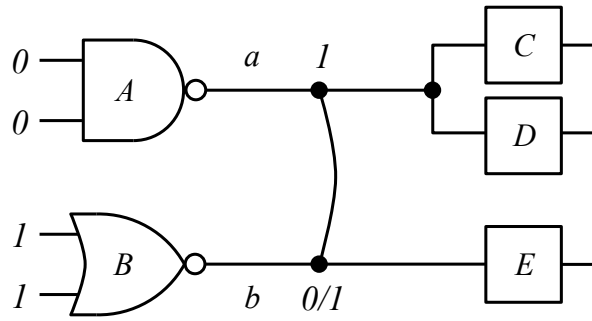


Figure 3.1: Example circuit with a bridging fault.

resistance or handle it implicitly. For both resistive and non-resistive models our discussion will be restricted to static testing methods.

Figure 3.1 illustrates the terms we just introduced. It depicts a circuit with a two-node inter-gate bridging fault creating an unwanted connection between nodes a and b . We will call the gates directly preceding the shorted nodes – here A and B – the *driving gates*. As A drives its output a to logical value 1 and B drives its output b to logical value 0, the bridge is *activated*. The gates directly succeeding the bridged nodes (C , D , and E) will be called *driven gates*. Typically only one input of each driven gate is directly connected to any of the shorted nodes. Thus, when appropriate, we will identify the driven gate with this input. We will distinguish the logical values imposed on the shorted nodes by the driving gates from the values actually observed by the driven gates. The former will be called *driven values* (in accordance with [2]). In this example, gate E is assumed to recognize the faulty logical value 1 instead of the fault-free driven value 0 due to the fault. Gates C and D interpret the fault-free driven value 1.

3.2 Bridging Fault Model Classification

There exists a large body of knowledge on the modeling of shorts. Yet, basically there is no such thing as *the* bridging fault model. Rather, this notion is a placeholder for a model which reflects the impact of a physical defect shorting conducting elements of a circuit. The characteristics of the actual model depend on many factors such as the technology used to implement the circuits under consideration, the desired modeling accuracy, and the computational complexity granted by the user. The latter is measured as the number of combinations of logical assignments to the driving gates, and fault effects seen by the driven gates, which is instantiated to represent a single bridging fault. The complexity is directly related to the computational effort required when performing fault simulation or test pattern generation using the respective model. Motivated by their physical nature, there exists to a certain degree a common understanding of certain properties of shorts which have to be respected by any bridging fault model. These properties are summarized in Chapter 3.1.

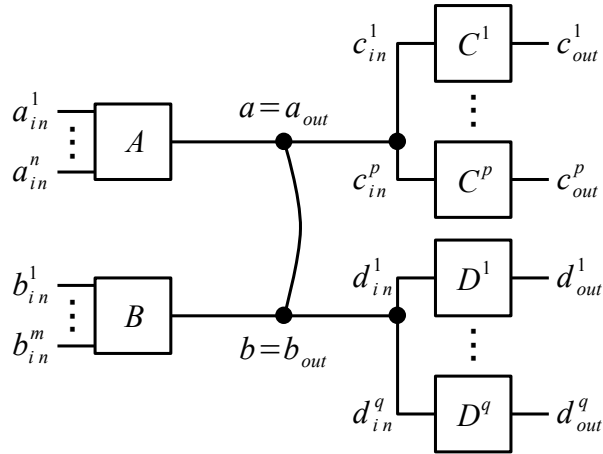


Figure 3.2: Portion of the circuit relevant for bridging fault models.

One of the parameters which recently was added to the list of relevant properties is the short's intrinsic resistance. It was found that this resistance heavily influences the electrical behavior of the defect and thus its impact on the logic function of the defective circuit. While some authors already account for this relevant parameter and factor it into their resistive bridging fault model, the majority of publications addresses non-resistive bridging fault models. The latter do not (explicitly) handle the intrinsic defect resistance. This means the models disregard that the exact resistance is dependent on the particular defect and cannot be known a priori. Non-resistive models either completely neglect the influence of this random parameter or assume an arbitrary but fixed value for the resistance.² There are also approaches which try to model circuit behavior for any (reasonable) defect resistance. This considerably reduces modeling complexity, although it comes at the expense of accuracy.

In fact, modeling accuracy is another important aspect. It is nearly indispensable that extremely accurate replication of a short's effects can only be achieved at the expense of high computational costs. Yet, given the large number of potential bridging fault locations, this is hardly acceptable for any practical application. It is a common understanding among most authors in the field that a sufficiently accurate replication can be obtained when focusing on very few gates surrounding the fault site. This localized view allows the use of classical Boolean techniques for fault simulation and test pattern generation – at least for the larger part of the circuit. The fault site itself can be evaluated by different techniques using several levels of abstraction. Usually most of the exact data generated at lower levels is mapped back to the gate-level, i.e. to Boolean values.

We will in the following limit our discussion to models considering the circuit excerpt depicted in Figure 3.2. The bridging fault affects two nodes a_{out} and b_{out} (for reasons of simplicity they will be referred to as a and b respectively). The shorted node $a(b)$ is driven by gate $A(B)$ which has $n(m)$ inputs $a_{in}^1, \dots, a_{in}^n (b_{in}^1, \dots, b_{in}^m)$. Node $a(b)$ is observed by $p(q)$

²A special case is the hard short [108], i.e. a defect whose intrinsic resistance is assumed to be (sufficiently close to) 0Ω .

gates $C^1, \dots, C^p (D^1, \dots, D^q)$. Each C^i , $1 \leq i \leq p$, has one output c_{out}^i and is connected to node a via input c_{in}^i . Each D^j , $1 \leq j \leq q$, has one output d_{out}^j . The gate's input d_{in}^j is connected to node b . Any further inputs to the driven gates have been omitted in the figure. This model of a bridging fault is the prototype used for the explanations in the chapters to come.

3.3 Bridging Fault Extraction

As we have hinted above it is impractical to test a circuit for all theoretically possible two-node shorts. This, however, does not necessarily imply that many defects have to remain undetected. Rather, it is important to focus on testing those shorts which are likely to be present in the circuit under consideration. Many of the shorts which are theoretically possible, actually only occur very rarely in a real circuit. Imagine for instance two interconnects which are physically located in opposite corners of the manufactured circuit. Theoretically these interconnects may be shorted by a defect. In practice, however, a defect affecting both interconnects rarely occurs. In fact a manufacturing imperfection which results in these two interconnects being shorted would severely impact the behavior of the circuit. Thus, it is likely that it will be detected by one of the numerous screening procedures conducted before the actual logic-level test.

Therefore, the question we have to ask is: Which shorts are actually likely to occur and should be tested for? The two most common techniques which try to answer this question are based on either the concept of critical area or the fringe capacitances. Both approaches first inspect the circuit layout to identify all pairs of circuit interconnects which are adjacent. In a second (optional) step these pairs of interconnects are sorted according to their likelihood of occurrence (in descending order). Finally each interconnect is mapped to the equivalent node in the gate-level netlist. Those pairs of nodes corresponding to the n topmost interconnect pairs are assumed to be the most probable candidates for a short. Threshold n may be set by the user depending on the desired coverage of defects. Since the resulting potential short locations are derived from the circuit layout we will refer to them as layout extracted shorts.

The most popular technique to extract probable short locations from a circuit layout is based on the *critical area* [46]. The critical area of two adjacent interconnects is defined as that part of their region of adjacency in which the center of a defect can lie in order to short the two interconnects. Those pairs of nodes corresponding to the n interconnect pairs with the largest critical area are assumed to be the most probable candidates for a short. It is evident that the size of the critical area is directly dependent upon the size of a potential defect. Thus, the defect size distribution [180] is a prerequisite for this metric. Unfortunately, this distribution may not always be available. Furthermore, computation of the critical area is a very time consuming task. As a consequence, the critical area metric may not always be applicable. Numerous algorithms and tools exploiting the concept of critical area have been developed [7, 44, 57, 61, 75, 125, 129, 130, 179, 195, 211, 213]. For the most part these approaches differ in the algorithm employed to compute the critical

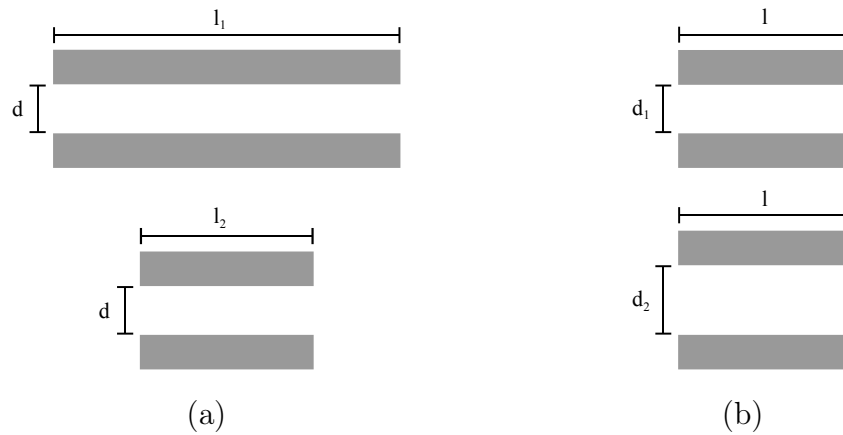


Figure 3.3: Influence of adjacency region and lateral distance on capacitance.

area and the assumptions made during that computation. Current implementations have evolved the algorithms for critical area extraction such that they can even be applied to large designs, like a commercial microprocessor [92].

Alternatively, layout extracted shorts can also be obtained by exploiting the *fringe capacitances* [110, 182]. The fringe capacitance is usually extracted during the construction of an integrated circuit. It is defined between two interconnects, and is composed of the coupling capacitances of all conducting elements belonging to either interconnect. Potential short locations are identified by extracting the fringe capacitances between all pairs of interconnects adjacent in the layout. Those pairs of nodes corresponding to the n interconnect pairs with the largest fringe capacitance are assumed to be the most probable candidates for a short.

The idea of this approach can be motivated by considering the parallel-plate model of capacitance (although in practice the fringe capacitance is computed using a far more elaborated model). In the parallel-plate model, capacitance C is expressed with respect to the dielectric constant ε , the area A of the plates, and the distance d between them. Given these parameters C can be calculated as $C = \varepsilon \cdot A/d$.³ This means that the capacitance between the two conducting elements is proportional to the area and inversely proportional to the distance between them. The very same relation also holds for the probability that a bridging fault occurs between two conducting elements. This is illustrated in Figure 3.3(a) for two pairs of conductors which both have the same lateral distance d . Their region of adjacency, however, is different. The upper pair of conductors runs in parallel for distance l_1 . As the lower pair only runs in parallel for distance l_2 , and $l_1 \gg l_2$, the upper pair of conductors has a higher capacitance. Analogously, the upper pair of interconnects would also have a higher probability to be affected by a short than the lower pair. In Figure 3.3(b) the region of adjacency of either pair of conductors is the same – both run in parallel for a distance l . Their lateral distance however, is different. As the lower pair of interconnects is further apart than the upper one ($d_1 \ll d_2$) the upper pair has a higher capacitance.

³Constant ε can be set to the dielectric constant of the layer on which the interconnects under consideration are located.

Analogously, again the upper pair of interconnects would have a higher probability to be affected by a short than the lower pair. This reasoning is very similar to the one used to motivate the critical area based metric – the relation between both metrics is further explored in [55].

Another way to obtain layout extracted shorts is discussed by Khare et al. in [84]. They propose to identify potential short locations by simulating the manufacturing process step by step. At each step, possible contaminations are injected and their potential to cause a short is evaluated.

We conducted all our experiments reported in this thesis for randomly extracted short locations. We decided not to use any layout extracted shorts, as this would have biased our results towards the specifics of a particular layout. This would only introduce another stochastic variable into the experimental setup which distracted from the essence of the respective experiment.

4 Non-Resistive Bridging Fault Models

This chapter gives a brief overview of non-resistive bridging fault models known in literature. These models differ substantially in terms of complexity and accuracy. The simplest approaches consider only the Boolean values at the two bridged nodes. They use basic Boolean functions to derive the logical values seen by the driven gates in the presence of a short. Examples are the dominance-behavior [202] and the wired-logic [115] models, as well as the closely related 4-way [92] model. More elaborate approaches, such as the precise test generation model proposed by Maeda et al. [107], try to account for any potentially possible defect induced behavior by exhaustively testing both driven gates. Alternatively, all possible stuck-at fault combinations are tested by the unified model introduced by our co-operation partners Chen et al. [P10]. The voting model by Acken et al. [3, 43, 121] targets accurate replication of the defect by including the driving gates' electrical properties and the assignments to their inputs into the modeling procedure. An extension by Maxwell et al. [112], called biased voting, additionally considers electrical parameters of each driven gate to improve the modeling accuracy. Some authors go beyond this by introducing sophisticated procedures which compute the impact of a short by using switch-level [35] or transistor-level [37] descriptions of the defect site. In [60], the bridging fault simulator E-PROOFS is proposed by Greenstein et al. which uses electrical simulations at run-time instead. Even though this approach suggests very high modeling accuracy, the computational overhead introduced by electrical simulations and complex matrix operations (“with the time complexity of $O(N^3)$, where N is the number of circuit nodes [59]” quoted in [100]) leads to prohibitive run-times for large circuits with many bridging faults. To alleviate this problem several authors, e.g. [27, 29, 146], recommend performing all necessary electrical simulations in a preprocessing step and to tabulate the results. This strategy maintains high modeling accuracy at reduced computation time. However, depending on the size of the gate library used to implement the circuit under consideration, these tables may become very large; improved data storage has been reported by Di et al. in [38]. Furthermore, tables may be inflexible with respect to changes of the gate library. Comparative studies covering some of the aforementioned non-resistive bridging fault models can be found in [12, 105].

In the following we will focus on models which go beyond the straightforward tabulation of data gained by electrical simulations. The spectrum of models to be discussed includes simple logic models, which offer reduced modeling accuracy at low computational costs (see Chapter 4.1). Chapter 4.2 will target technology-based models, merging low-level technological knowledge (similar to that gained from electrical simulations) with a compact and efficient data representation to form an accurate and simple modeling strategy. Generalizing approaches are introduced in Chapter 4.3. They use an abstracting description of

the changes in logic behavior caused by a short to avoid biasing the model towards specific technological parameters. This, however, leads to high computational overhead.

4.1 Simple Logic Models

The simple logic models are directly related to the single-stuck-at fault model. They neither explicitly consider any low-level technical data, nor do they try to account for low-level effects by using a generic description of the defect induced behavior. Chapter 4.1.1 introduces the well-known dominance-behavior and wired-logic models. Both models are combined by the 4-way model which recently gained popularity and will be discussed in Chapter 4.1.2.

4.1.1 Wired-Logic And Dominance-Behavior Models

The early approaches known for modeling the logic effects caused by a short were actually extensions of the single-stuck-at fault model. Similar to the latter model, it is assumed that the activation of a fault can be determined locally by evaluating the driven values at the fault site in the good circuit. Yet, in contrast to the single-stuck-at case, two nodes are affected by the bridge and thus the logic state of both nodes has to be considered. Moreover, fault effects of an activated bridge are assumed to be observable by all successors of one of the two bridged nodes only. Again this is comparable to a stuck-at fault affecting the stem of this node. This assumption is justified by the observation that in certain logic families the two driving gates are not equally capable of affecting the electrical state of the bridged nodes (this will be discussed below in more detail). Usually, one of the driving gates is more influential such that the logical value driven at its output node, called the *aggressor node*, overrides the logical value driven at its counterpart, the *victim node*. In the presence of a short, the gates connected to the victim node interpret the logical value imposed by the aggressor node. Depending on the criteria for assigning the roles of aggressor and victim to the shorted nodes, and the way in which the faulty logical value is determined, the early bridging fault models can be divided into two classes: the *dominance-behavior* [202] and the *wired-logic* [115, 161] models.

The dominance-behavior models attribute to each node involved in a bridge a fixed role. Given a bridge between two nodes a and b , either a or b can be the aggressor, i.e. the dominant node. Two combinations of assigning aggressor and victim to a and b are possible. Hence, there are two different dominance-behavior models: *a-dominant* and *b-dominant*.

In contrast to the dominance-behavior models in the wired-logic models, the assignment of the roles of aggressor and victim is done with respect to the logic state of the bridged nodes. There are two different versions, one of them being the *wired-AND* model. In this model the bridged node driven to logical value 0 in the fault-free circuit is the aggressor, whereas its counterpart driven to logical value 1 is the victim. The victim's successors interpret the faulty logical value 0. In other words the value interpreted by all driven gates

Table 4.1: Aggressor and victim nodes for wired-logic and dominance-behavior models, “X” marks combination of aggressor, victim, and stuck-at fault used by the model.

Value at		Aggressor	Victim		wired-logic		dominance-behavior	
a	b	Node	Node	Fault	wired-AND	wired-OR	a -dom.	b -dom.
1	0	a	b	s-a-1	-	X	X	-
		b	a	s-a-0	X	-	-	X
0	1	a	b	s-a-0	X	-	X	-
		b	a	s-a-1	-	X	-	X

corresponds to the logical AND of the logic assignments seen in the fault-free circuit at the bridged nodes.

In a similar manner the second variant of the model, called *wired-OR*, can be defined by attributing the role of the aggressor to the node driven to logical value 1. This corresponds to calculating the logical OR of the values assigned to the nodes involved in the bridge.

Table 4.1 summarizes the models assuming a bridging fault between two nodes a and b . The first two columns state the driven values for the bridged nodes (only combinations activating the fault are considered). The third column identifies the aggressor node. Victim node and stuck-at fault to be inserted at its stem are stated in columns four and five, respectively. Depending on the logic state of a and b , each model uses a different combination of aggressor/victim assignment and stuck-at fault to represent the behavior of the bridge. This mapping is given in columns six to nine. As we can see the conjunction of both fault models in either class results in the same set of combinations of aggressor, victim, and stuck-at fault.

Relation to Stuck-At faults

The interdependence between stuck-at faults and the wired-logic and dominance-behavior bridging fault models has been the subject of several publications (see e.g. [1, p. 292ff.] for an overview). Mei [115] extensively studied the detection conditions for various types of bridging faults and their relation to stuck-at faults. Friedman [47] sought to extend stuck-at fault test pattern generation strategies such that the generated test patterns detect certain classes of wired-logic bridging faults as well. Kodandapani et al. [87] developed the concept of redundancy for bridging faults and demonstrated that a redundant bridging fault can render tests for stuck-at faults invalid. The approach by Abramovici et al. [2] allows one to derive the coverage of bridging faults from stuck-at fault simulation and discusses test pattern generation for less restricted classes of bridging faults.

All studies exploit one key property: In principle the early bridging fault models inject a s-a- v fault at the victim node (which has to be activated by assigning \bar{v} to the node) and additionally enforce logical value v on the aggressor node. This has been, for instance, reported by Williams et al. [202]. For the wired-AND model their findings can be formalized as follows (adapted from [2]):

A test t detects the wired-AND non-feedback bridging fault between nodes a and b iff either t detects a s-a-0 and sets $b = 0$, or detects b s-a-0 and sets $a = 0$.

This only holds if the stuck-at faults affect the stem of a node, i.e. all successors of this node. A detailed proof can be found in [1, p. 292]. Obviously it is possible to make similar assertions for wired-OR, a -dominant, and b -dominant models.

Technological Background

The close relation of both wired-logic and dominance-behavior model to stuck-at faults (which hardly refer to real physical defects) might be somewhat misleading. In fact these bridging fault models are motivated by the impact of actual shorts in certain logic families.

If two nodes are shorted in a circuit implemented in *emitter-coupled logic* (ECL), the one with driven value 1 will always override the node carrying logical value 0. In this technology the abovementioned justification for the basic bridging fault models holds: A short can indeed be modeled as a wired-OR [116]. To *n-type metal oxide semiconductor logic* (NMOS), the contrary applies, i.e. logical value 1 will always be overridden by logical value 0 in presence of a short. For this logic family wired-AND is a valid bridging fault model [121]. Similar arguments justify using the wired-AND model for circuits implemented in *transistor-transistor logic* (TTL) [87]. Two nodes shorted in a circuit implemented in *complementary metal-oxide-semiconductor* (CMOS) digital logic may see voltage levels which cannot be interpreted unambiguously by all driven gates. Therefore the wired-logic and dominance-behavior models do not apply in this case [4]. This has been demonstrated by electrical simulations in [43, 60, 121]. In general, circuits are implemented using only one logic family. Hence, there is no technological justification for considering a mixture of any of the bridging fault models introduced so far [115].

We will in the following focus on the CMOS digital logic family. Even though wired-logic and dominance-behavior bridging fault models have no technological foundation in this technology, they still provide a good example of a simple and technologically sound bridging fault model.

4.1.2 The 4-way Model

When modeling bridging faults in an industrial setting it is not only the model's accuracy that matters. Excessive run-times of ATPG and simulation tools for multi-million gate designs cannot be afforded. Therefore the model's computational complexity should be moderate. Compatibility with standard commercial tools typically developed for single-stuck-at faults is favorable as well. Furthermore, certain technological data might not be accessible (e.g. for reasons of intellectual property protection). A high-level approach, only relying on gate-level information, might be the only bridging fault model applicable in such cases.

Table 4.2: Aggressor and victim nodes for 4-way model.

Value at		Aggressor	Victim	
a	b	Node	Node	Fault
1	0	a	b	s-a-1
		b	a	s-a-0
0	1	a	b	s-a-0
		b	a	s-a-1

All the abovementioned requirements are fulfilled by the wired-logic and dominance-behavior models. But as each of them has been demonstrated to be inappropriate for CMOS digital logic (see Chapter 4.1.1), it is questionable whether their application in an industrial setting will help to achieve low DPM targets. Findings on the interdependence of wired-OR, wired-AND, a -dominant, and b -dominant models in [105], however, indicate that the combined use of both wired-logic (or both dominance-behavior) models may mitigate the disadvantages of each individual model.

One of the first references for the industrial application of the *4-way model* is [92]. It reports on a bridging fault model which represents each fault by four combinations of aggressor constraint and victim stuck-at fault which has been applied to a commercial microprocessor. Further studies from the same context can be found in [25]. Some authors refer to the 4-way model as to the *Aggressor/Victim model* [102].

Table 4.2 states constraints and single-stuck-at faults assigned to aggressor and victim nodes by the 4-way model (assuming a bridge between two nodes a and b). The driven values on these nodes are specified in columns one and two. Column three(four) denotes the aggressor(victim) node. The stuck-at fault assigned to the victim node is given in column five. The 4-way model only adds a constraint on the aggressor node to the single-stuck-at fault model. Thus, handling of 4-way faults by (modified) state-of-the-art stuck-at tools is possible.

Cusey et al. [34] proposed the use of an extended version of the 4-way model for their automatic test pattern generator BART. Their approach, targeting the CMOS technology, combines the basic 4-way model with electrical awareness. BART seeks to assign logical values to the inputs of the driving gates which (due to the properties of CMOS logic gates) enhance the probability that faulty logical values actually occur at the node predicted by the 4-way model. In Chapter 8.2 we compare experimental results obtained with our tool RBF-ATPG with data presented in [34]. The experimental results for the 4-way model can be found there as well.

4.2 Technology-Based Models

The simple logic models presented so far do not exploit any low-level knowledge about the functionality of CMOS digital logic. In this respect it is questionable if they can

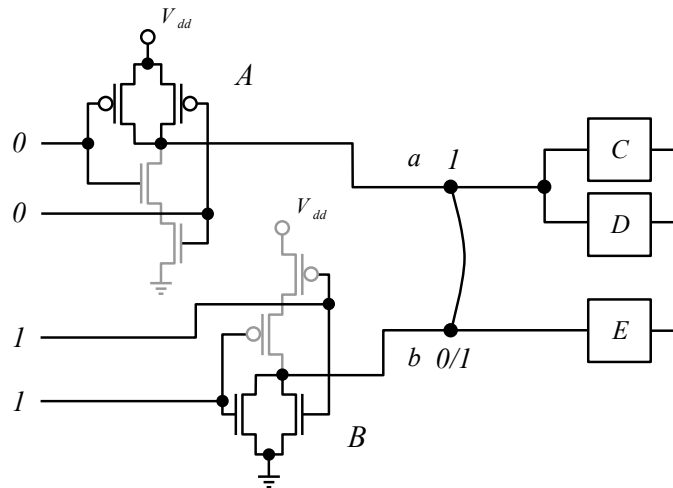


Figure 4.1: Transistor level view of driving gates in CMOS technology.

accurately reflect the effect of a bridging fault in this logic family. For the wired-logic and dominance-behavior models simple electrical simulations did show that this is not the case. In the following we will introduce two models which have been developed specifically for CMOS technology.

At first, however, we will briefly discuss the impact of a short in a circuit implemented in CMOS technology. Such a defect, shorting the outputs of two gates, connects the transistor networks within the gates which were originally designed to drive two (electrically) separate nodes. The capacitive load the networks have to drive may be approximated by the sum of the lumped loads connected to each individual gate output (further load is e.g. contributed by capacitive coupling etc.). If the bridge is activated, one of the gates tries to drive its output to logical value 1, i.e. its pull-up network is activated. The other gate attempts to drive its output to logical value 0, which means that its pull-down network is active. This is illustrated in Figure 4.1 which depicts a short between two gate outputs a and b (this is the same situation as in Figure 3.1). Both driving gates are shown at the transistor level – node a is driven by NAND gate A with two inputs, node b is driven by the two-input NOR gate B . The dominant part of the (cumulative) capacitive load is contributed by the transistors in the driven gates C , D , and E . Logical assignment $(0, 0)$ to the inputs of gate A activates both p -transistors connected in parallel in the pull-up network of that gate. The pull-down network consisting of two n -transistors connected in series is inactive. Logical assignment $(1, 1)$ applied to gate B activates both n -transistors connected in parallel in the gate's pull-down network. At the same time both p -transistors connected in series are inactive. Due to the short, the current supplied by the pull-up network in A to charge the load is drained immediately by the discharging pull-down network of B . The current left to charge the cumulative capacitive load is – according to *Kirchhoff's first law*¹ – equal to the difference of the currents which flow through the active transistor networks. As this current is lower than that in a fault-free node, the transistors in the succeeding (driven)

¹The sum of the currents flowing into a node is equal to the sum of the currents flowing out of the node.

gates may not be charged to full-scale voltage levels. Consequently, a closer examination is necessary to determine the logical values interpreted by these gates.

Acken et al. proposed the voting model in [3]. The model uses a strategy which allows it to compute the logical values interpreted by the driven gates. Its essence is the combination of profound technological data with a voting scheme. This allows it to resolve digital values in a very simple manner. We introduce the voting model in Chapter 4.2.1. It has been demonstrated, however, that some of the simplifying assumptions used by the voting model may affect modeling accuracy. An extension of the model, the *biased voting model*, proposed by Maxwell et al. [112], helps to alleviate these problems. This model will be covered in Chapter 4.2.2.

4.2.1 The Voting Model

According to the paradigm of the *voting model* [3, 4, 120], the driving gate whose transistor network is capable of providing or draining more current will determine the state of the capacitive load and thus the logical values recognized by the driven gates. If for instance the p -transistor network can sustain more current than the n -transistor network is able to drain the voltage potential of the bridged nodes will be closer to V_{dd} and the driven gates are assumed to interpret logical value 1.

The amount of current flowing through a gate is determined by two major factors. On the one hand, there are static design parameters such as the topology of the transistor networks (connection in parallel or in series), process parameters (such as doping etc.), and the size of each transistor. The latter is defined by the width (W) and length (L) of the transistor's gate, subsumed by the W/L ratio. The amount of current flowing through a transistor is proportional to its W/L ratio. On the other hand the current is determined by the logical assignment to the gate's inputs which is dependent on the current logic state of the whole circuit and is thus variable. The assignment influences not only which transistor network is conducting but also which of the transistors within this network are active. An example is depicted in Figure 4.1 where (0, 0) assigned to gate A activates two p -transistors. For a given gate and an input assignment the voting model represents all factors impacting the ability of the active transistors to drive or drain current by the *equivalent resistance* R . This resistance is inversely proportional to the amount of current conducted by the active transistors. More intuitively, this is captured by the active transistors' *conductance* C which is defined as $C = \frac{1}{R}$. When normalized by the conductance of a single n -transistor, the conductance of the active p - or n -transistors is called *relative puissance* [120], denoted by RP .² The relative puissances can be obtained by electrical simulations. For a given logic gate library, all transistor networks used in the logic gates are simulated for all possible input assignments. The relative puissances thus obtained are tabulated. Let $c_p(c_n)$ be the number of legal configurations of the $p(n)$ -transistor networks for the library. Hence, the size of this table will be $c_p \cdot c_n$. Assuming the library of the circuit in Figure 4.1 contains

²The term "puissance" is – according to [4] – supposed to underline that due to the non-linear properties of transistors the normalized conductance can neither be used to determine voltages nor time constants.

only a two-input NAND and a two-input NOR we obtain the following legal transistor configurations for both pull-up and pull-down network: single transistor, two transistors in series, and two transistors in parallel. As a consequence the table of relative puissances for this circuit would have nine entries.

According to the voting model, the relative puissances of the complementary transistor networks driving the shorted nodes may be compared to determine the logical value observed by the driven gates. Given a bridging fault and an activating assignment to the driving gates this can be done as follows:

- (1) Identify the active transistors in both the pull-up and pull-down network. Fetch the relative puissance RP_u (RP_d) of the active transistors in the pull-up (pull-down) network from the precomputed table of relative puissances.
- (2) Compare the relative puissances: If $RP_u > RP_d$ the p -transistor network is able to conduct more current and consequently logical value 1 is interpreted by the driven gates. Otherwise the n -transistors are dominant and logical value 0 is interpreted.

Step (1) of this procedure first identifies which of the transistors actually drive the shorted nodes according to the input assignment. Then the corresponding relative puissances are determined. Step (2) of this procedure is similar to a vote. The driving gate with the larger RP “wins” and all driven gates are assumed to interpret the “winner’s” driven value. In the situation depicted in Figure 4.1 the relative puissance RP_u of the pull-up network of gate A is larger than the relative puissance RP_d of the pull-down network of gate B . Hence, the voting model would predict that the pull-up network wins and therefore all driven gates interpret logical value 1.

So far we have presumed that the voltage induced at the shorted nodes can always be interpreted as a well-defined logical value by the driven gates. However, as confirmed by Acken et al. in [3] this voltage is unlikely to reach the extreme values ground and V_{dd} , respectively. Rather, it will assume some intermediate level which does not necessarily have to be resolvable unambiguously to a logical value. Experiments conducted by Acken et al. indicate that only very few driving gate configurations may result in a voltage at the bridged nodes which falls into this critical range. This motivates the following conclusion: “a short in that range is unlikely” [3]. Thus, assuming the interpretation of a well-defined logical value is mostly correct.

4.2.2 The Biased Voting Model

Like any other bridging fault model, the voting model as presented in Chapter 4.2.1 uses several assumptions to reduce modeling complexity. However, as highlighted by Maxwell et al. in [112] some of these assumptions can have a detrimental impact on the model’s accuracy. To overcome this deficiency they proposed the *biased voting model* [112] which refines some assumptions introduced in the voting model while at the same time maintaining its simplicity.

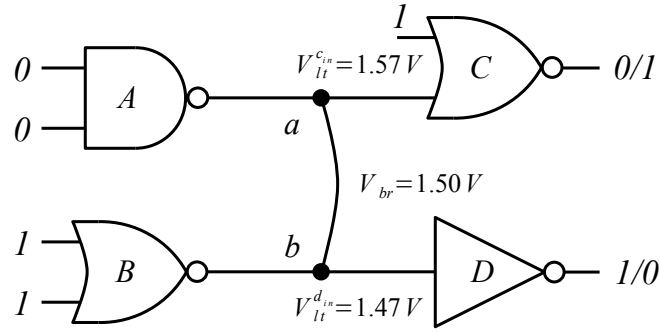


Figure 4.2: Interpretation of voltage V_{br} at the bridged nodes a and b by driven gates C and D .

The first and most severe inaccuracy they identified is caused by the handling of logic threshold voltages. The voting model assumes a single fixed threshold for all inputs of all gates ($V_{dd}/2$ in [3]). Yet as outlined in Chapter 2.1.2, the logic thresholds of different gate types and inputs are not necessarily equal. Hence, using a single fixed value may lead to wrong predictions as illustrated in Figure 4.2. Assume that voltage V_{br} induced on the bridged nodes a and b is 1.50 V (for $V_{dd} = 3.30\text{ V}$). As the logic threshold V_{lt}^{cin} of gate C is 1.57 V and thus larger than V_{br} the gate interprets logical value 0. Whereas the fault-free logical value driven by A is logical value 1. According to the logic threshold $V_{lt}^{din} = 1.47\text{ V}$ of gate D this gate interprets logical value 1 – instead of the fault-free logical value 0 driven by B . If we had considered a single fixed logic threshold voltage of $V_{dd}/2 = 1.65\text{ V}$, only gate C would be assumed to interpret a faulty logical value, i.e. logical value 0. Obviously not predicting a faulty logical value when actually a difference to the fault-free behavior could be observed may result in an underestimation of the fault coverage. Inversely, the model could also predict faulty logical values to be observable which actually are not present. When different gates interpret the same voltage level as contrary logical values, this is called *Byzantine General’s Problem* [4].³

For the voting model the Byzantine General’s Problem has been solved in a refined version of the model introduced in [4]. In addition to that, the biased voting model is able to tackle a second source of inaccuracy which is inherent to the voting model. The relative puissances of the latter model are determined for a fixed voltage V_{br} at the bridged nodes ($V_{br} = V_{dd}/2$ is used in [3]). As demonstrated by Maxwell et al. in [112] this overly simplifies the matter. The relative puissance of a transistor (network) is defined as the conductance of this network relative to the conductance of a single n -transistor. Electrical simulations clearly indicate that the conductance of a transistor (network) changes with varying V_{br} . Let $C_p(C_n)$ be the conductance of a $p(n)$ -transistor network. The experiments show that compared to their value at $V_{br} = V_{dd}/2$ for $V_{br} > V_{dd}/2$ the value of $C_p(C_n)$ is increasing(decreasing). Contrary to that for $V_{br} < V_{dd}/2$ the value of $C_p(C_n)$ is decreasing(increasing). Data in [112] proves that this effect heavily impacts the relative puissances and thus the outcome of the vote. In conclusion, in order to obtain accurate results it is inevitable to consider the conductance of each transistor network with respect to the voltage induced at the bridged

³This alludes to the “Byzantine Generals Problem” known in communication theory (see [96]).

nodes. Unfortunately, if we want to correctly handle the Byzantine General’s Problem as well, this leads to the following dilemma: To determine the logical value interpreted by a gate we have to know V_{br} (or at least its relative magnitude with respect to the gate input’s logic threshold voltage). It is possible to calculate V_{br} if the conductances of the transistor networks driving the bridged nodes are known. These are, however, dependent on V_{br} .

Maxwell et al. proposed a procedure to break this “vicious circle”. It is based on the assumption of an idealized circuit in which the conductances of both pull-up and pull-down network are independent of V_{br} . The procedure has to be applied individually to every input of all driven gates connected to either of the shorted nodes. Let V_{lt} be the logic threshold of such an input:

- (1) Determine the ratio r_i of the conductance of the pull-down and pull-up network for $V_{br} = V_{lt}$ assuming an idealized circuit.
- (2) Identify the active transistors in both the pull-up and pull-down network. Then determine the ratio r_t of the conductances of the pull-down and pull-up network the actual transistors would require in order for $V_{br} = V_{lt}$ to hold.
- (3) Compare r_i and r_t : If $r_t > r_i$, logical value 0 will be interpreted by the driven gate; otherwise the gate will see logical value 1.

Step (1) of this procedure determines the ratio r_i of conductances which would be required for an idealized circuit to attain $V_{br} = V_{lt}$. In the next step (2) the ratio r_t of conductances of the actual active transistor networks is computed again assuming $V_{br} = V_{lt}$. Typically this data has been precomputed by electrical simulations and can now be retrieved from tables. If $r_t > r_i$ the pull-down network of the actual transistors is much stronger than necessary to induce $V_{br} = V_{lt}$ (see step (3)). Therefore, the pull-down network “wins” the vote. Inversely, if $r_t < r_i$ the pull-down network of the actual transistors is weaker than required to pull the voltage below V_{lt} . Consequently, in this case the pull-up network “wins” the vote.

In summary, the biased voting model is able to correctly handle the Byzantine General’s Problem and further increases modeling accuracy by considering the voltage dependency of transistor conductance. At the same time the complexity of the voting procedure, introduced by the voting model, is only marginally increased.

4.3 Generalized Logic Models

In the preceding chapters we have highlighted that several parameters impact the effects caused by a short. A bridging fault model addressing as many of them as possible can be expected to be very accurate. However, we have also emphasized that some of these parameters may be inaccessible in certain application scenarios. Additionally, parametric variations expected to occur in future deep sub-micron technologies might introduce further complexity which might complicate accurate (technology-based) models. The generalized logic models address these issues by using a generic description of a short’s effects – thus

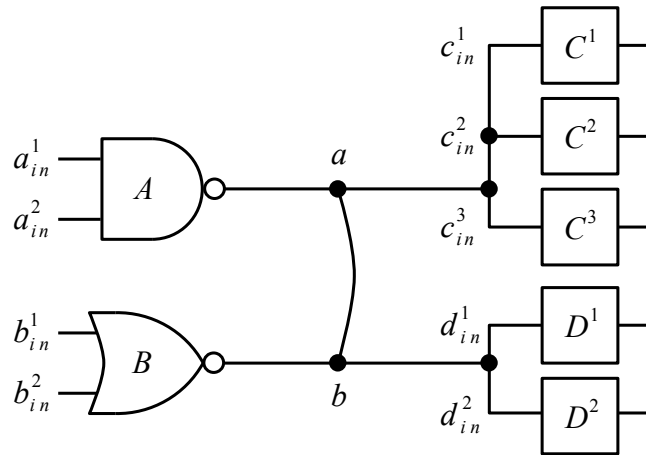


Figure 4.3: Example of circuit with a non-resistive bridging fault.

avoiding the need for low-level circuit parameters. In the following we will introduce two representatives of this type: The unified model (Chapter 4.3.1) and the precise test generation model (Chapter 4.3.2).

4.3.1 The Unified Model

The *unified model*⁴, developed by our co-operation partners Chen et al. [P10] pursues a completely different approach than the technology-based models from Chapter 4.2. Instead of trying to bring the accuracy of bridging fault modeling to perfection, it attempts to capture the behavior of a bridge by a generic description of all potential fault effects caused by a short.

The electrical state of two nodes affected by a short is mainly influenced by the relative strengths of the driving gates and the defect's resistance. Because of the Byzantine General's Problem, each driven gate observing one of these nodes may interpret its state differently depending on the gate input's logic threshold. Consider a defect shorting two nodes a and b (see Figure 4.3) and five driven gates C^1 , C^2 , C^3 , D^1 , and D^2 . Given a fixed assignment to a and b which activates the bridge each driven gate either interprets the fault-free state of the respective bridged node or it determines the faulty (inverse) value. For five gates there are $2^5 = 32$ combinations of these two states. Omitting the state in which all gates interpret the respective fault-free logical value, we can have $2^5 - 1 = 31$ combinations of at least one faulty logical value. This is equivalent to assigning all possible multiple-stuck-at faults to the gate inputs c_{in}^1 , c_{in}^2 , c_{in}^3 , d_{in}^1 , and d_{in}^2 . In general, for a bridge shorting node a with p branches and node b with q branches (and assuming a fixed activating assignment) we obtain $2^{p+q} - 1$ multiple-stuck-at faults. Note that there might be multiple assignments to the driving gates each activating the bridge. For every assignment the same number of

⁴The fault representation technique employed by the unified model can be applied to interconnect open defects as well. Hence, the model represents a unified approach which is able to handle both opens and bridges. Aspects of the model relating to open defects are covered in [148].

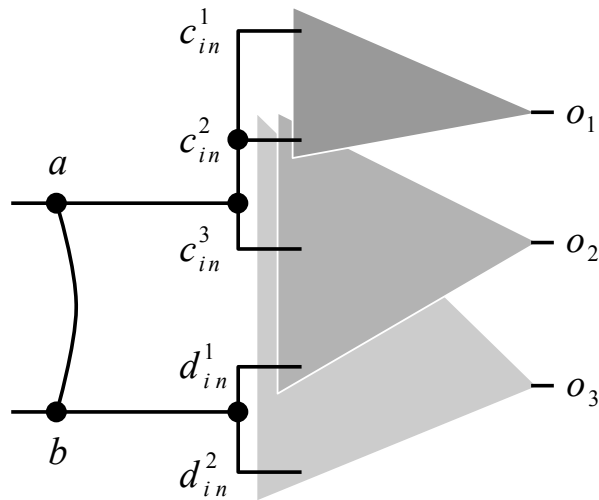


Figure 4.4: Primary outputs reachable from driven gates' inputs.

multiple-stuck-at faults is considered. In our example there are 10 activating assignments to the inputs of the two input NAND gate A and the two input NOR gate B . Obviously each logical assignment to the driving gates can be interpreted as a constraint. From this combination of a constraint and a multiple-stuck-at fault we can compose a CMS@ fault. Hence, we obtain $10 \cdot (2^5 - 1) = 310$ conditional multiple-stuck-at faults modeling the potential behavior of the bridge in Figure 4.3.

It is very time consuming to process this huge amount of CMS@ faults required to model a practically relevant number of bridges in a given circuit. In [P10] two strategies have been proposed to reduce the number of CMS@ faults which have to be addressed explicitly. The first strategy is called *implicit fault simulation*. Consider a s-a-1 fault at the input c_{in}^1 of C^1 in Figure 4.3. Assume furthermore that for a given assignment to gates A and B we can detect this fault at an observable point while at the same time applying an unknown value to the remaining driven gates, i.e. $c_{in}^2 = c_{in}^3 = d_{in}^1 = d_{in}^2 = X$. From this we can conclude that c_{in}^1 s-a-1 can be detected irrespective of the values interpreted by the other driven gates involved in the bridge. As a consequence we can mark all $2^4 = 16$ CMS@ faults involving c_{in}^1 s-a-1 as detected and remove them from the list of faults. Applying this strategy iteratively to each driven gate for all activating assignments allows to determine the detections statuses of large parts of the set of CMS@ faults.

The second strategy, called *structural analysis method*, exploits the notion of fan-out cones (i.e. the set of observable points reachable from a given circuit node via a path). Assume that (as depicted in Figure 4.4) in our example, fault effects at c_{in}^1 and c_{in}^2 can be detected at observable point o_1 . Furthermore c_{in}^2 , c_{in}^3 , and d_{in}^1 (c_{in}^2 , c_{in}^3 , d_{in}^1 , and d_{in}^2) can be observed at o_2 (o_3). A stuck-at fault affecting c_{in}^1 can only be detected at o_1 whereas the effect of a stuck-at fault at d_{in}^2 can solely be observed at o_3 . As there is no interaction between these two stuck-at faults during effect propagation, it is sufficient to either detect the fault at c_{in}^1 or the fault at d_{in}^2 to detect the bridging fault. Consequently it is not necessary to explicitly consider CMS@ faults describing multiple-stuck-at faults which affect both c_{in}^1

Table 4.3: Instances of unified fault model.

Affected Bridge Nodes	Activation	
	Implicit	Explicit
Single Faulty Node	ISFN	ESFN
Two Faulty Node	ITFN	ETFN

and d_{in}^2 . Based on this conclusion we can split the set of CMS@ faults describing this bridge into three groups: (1) those faults affecting c_{in}^1 and c_{in}^2 which can be detected at o_1 , (2) those CMS@ faults addressing c_{in}^2 , c_{in}^3 , and d_{in}^1 observable at o_2 , and (3) the group of faults involving c_{in}^2 , c_{in}^3 , d_{in}^1 , and d_{in}^2 detectable at o_3 . Note that the set of CMS@ faults contained in group (2) is a subset of those belonging to group (3). Thus we have to explicitly consider the CMS@ faults belonging to groups (1) and (3) only. In our example this reduces the number of CMS@ faults from $10 \cdot (2^5 - 1) = 310$ to $10 \cdot ((2^2 - 1) + (2^4 - 1)) = 180$. If we finally take into account that faults affecting c_{in}^2 are contained in both sets we can further reduce this number to 170 CMS@ faults. This is far less than the 310 faults contained in the original fault list.

The unified model allows us to model shorts at several levels of accuracy. By modulating the number of CMS@ faults used to describe a bridging fault, the modeling accuracy and thus the models complexity can be controlled. Given a bridge between two nodes a and b , it is possible that (some of) the gates driven by node a interpret a faulty logical value whereas all gates driven by node b see a fault-free logical value or vice versa. This type of bridging fault is referred to as *single faulty node* (SFN) bridge. On the other hand, the bridge could also affect both nodes such that gates driven by node a as well as gates driven by b may recognize a faulty logical value. This is termed a *two faulty node* (TFN) bridge. Bridging faults which induce different faulty logical values depending on the assignments to the driving gates are said to require *explicit activation*. For these faults, the unified model considers all assignments to driving gates activating the bridge. If, however, only the activation of the bridging fault is relevant – i.e. the logical values driven at the shorted nodes – the bridge is said to require *implicit activation*. In this case, all activating assignments to the driving gates producing the same pair of driven values are treated as equivalent. This is similar to the wired-logic and the dominance-behavior models covered in Chapter 4.1.1.

Based on these categories four different instances of the unified model can be defined. SFN(TFN) bridges requiring explicit activation are captured by the ESFN(ETFN) instance. Analogously, SFN(TFN) bridges for which implicit activation is sufficient are modeled by the ISFN(ITFN) instance. Table 4.3 summarizes the instances of the unified model; columns two and three refer to the activation, while rows two and three state the affected nodes. From the construction of the model's instances one can deduce that the set of CMS@ faults modeling ISFN bridges is contained in the set of faults representing ITFN bridges. Similarly the set of CMS@ fault modeling ESFN bridges is contained in the one created for ETFN bridges.

For the bridge example depicted in Figure 4.3 the number of CMS@ faults (without using any of the reduction techniques described above) ranges from $2 \cdot ((2^2 - 1) + (2^3 - 1)) = 20$ for the ISFN instance⁵ to $10 \cdot (2^5 - 1) = 310$ for the ETFN instance.

In an application scenario in which the relative order of the driven gates' logic thresholds is known (exact values are not required), a bridging fault can be modeled by a reduced number of CMS@ faults. Let V_{lt}^i be the logic threshold of gate input i . Assume that $V_{lt}^{c_1^1} < V_{lt}^{c_2^2} < V_{lt}^{c_3^3}$ and that the observed node a is driven to logical value 1 (see Figure 4.3). For gate C^1 to recognize a faulty logical value 0 the voltage at node a , V^a , has to be lower than the logic threshold of c_1^1 , i.e. $V^a < V_{lt}^{c_1^1}$. As $V_{lt}^{c_1^1} < V_{lt}^{c_2^2} < V_{lt}^{c_3^3}$ it is evident that in this case c_2^2 and c_3^3 have to recognize a faulty logical value as well. Exploiting this knowledge we can now eliminate all CMS@ faults which attribute a stuck-at-0 fault to c_1^1 but not to both c_2^2 and c_3^3 . Instances of the unified model which extend this reasoning to all driven gates, fault locations, and values are indicated by the suffix '_TH' (which alludes to the consideration of the logic threshold), e.g. ISFN_TH.

Our co-operation partners have successfully integrated the unified model into a state-of-the-art commercial tool for stuck-at test pattern generation and simulation. Moreover, they have applied the model to industrial circuits with more than one million gates. Further details and experimental results can be found in [P10]. In Chapter 8.2 we compare results on test pattern generation for the unified model from [P10] with data obtained with our tool RBF-ATPG.

4.3.2 The Precise Test Generation Model

The *precise test generation model* (PTG) by Maeda et al. [107], and the unified model (described in Chapter 4.3.1) share a very similar motivation. Both approaches try to implicitly model all potential electrical effects influencing the behavior of a bridge and both use a comparable, generalized description of this behavior. In the following we point out the key features of the PTG model and its differences with respect to the unified model. We will exclude, however, all aspects related to test generation. For details on this topic please refer to [107].

Just as the unified model, the PTG model neither addresses the relative strengths of the driving gates nor the non-predictable resistance of the intrinsic short resistance. Instead all assignments to the driving gates which activate the bridge are considered – the bridge is said to require explicit activation in the terminology of the unified model. Given a bridge between two nodes a and b and a fixed activating assignment, the PTG model assumes that the state of node a and the state of node b can be degraded by the defect. Yet the short cannot affect the successors of both nodes at the same time. This corresponds to a single faulty node (SFN) bridge in the nomenclature of the unified model. Contrary to the latter model the PTG approach does not consider the Byzantine General's Problem – all driven gates connected to the same bridged node are assumed to interpret the same (faulty) logical

⁵We assume that logical values on node a and on node b are (independently) degraded by the short.

value. Consequently, for a given activating assignment (equaling a constraint) the bridge is modeled by two CMS@ faults, one affecting node a and one addressing node b . The very same pair of faults is attributed to all remaining activating assignments. Assuming t activating assignments, the number of CMS@ faults used by the PTG model to describe a bridge is $2 \cdot t$. In contrast to the unified model, no techniques to reduce the number of faults are proposed for the PTG model in [107].

In summary, the PTG model can be best compared to the ESN instance of the unified model. Yet, as it does not provide for the Byzantine General's Problem the CMS@ faults used by the PTG model are only a subset of those encompassed by the ESN model. Consequently, the number of faults is reduced substantially at the expense of modeling accuracy.

5 The Resistive Bridging Fault Model

As we have pointed out in Chapter 4, the majority of bridging fault models address non-resistive faults. Yet, as it has been shown experimentally in [158], a substantial fraction of shorts have a non-zero resistance, typically lower than $500\ \Omega$ (this is also supported by data from [134]). The exact intrinsic resistance of this type of defect is dependent on many factors including parameters like its shape, size, conductivity, exact location on the die, evaporation behavior, and electromigration. Consequently, different shorts will have a different intrinsic resistance even if they may be represented by the same (resistive) bridging fault, i.e. affect the same pair of circuit nodes. When modeling resistive shorts we therefore have to treat their intrinsic resistance as an unknown parameter and should consider the whole continuum of this parameter.

A model which is able to efficiently and accurately reflect the behavior of a circuit in the presence of a short with random resistance is the resistive bridging fault model proposed by Renovell et al. in [152, 154]. In this chapter we will introduce this model. In doing so, we will demonstrate that the short's resistance has indeed a non-negligible influence on the behavior of a circuit, and that it should be modeled explicitly. It will turn out that a test vector which detects a short with one particular resistance might not be able to detect a short with another resistance even though both affect the same circuit nodes. Furthermore, we will demonstrate that the parametric nature of the resistive bridging fault model imposes fundamental changes on the meaning of well-known testing concepts such as fault coverage and redundancy.

In the following we will first elaborate on the influence of the intrinsic short resistance on circuit behavior and introduce the model as proposed by Renovell et al. in Chapter 5.1. The electrical aspects of the model will then be discussed in Chapter 5.2. In particular, we will give general indications in which way we can adapt the electrical core of the model to future technologies. We also report on how we successfully transferred the electrical core to two new technology nodes. Then, we will explain how Renovell et al. fitted the notion of fault detection to the requirements of a parametric fault model in Chapter 5.3. Probabilistic fault coverage definitions which reflect coverage of shorts with random resistance will be discussed in Chapter 5.4. Subsequently, in Chapter 5.5, we will bring these components together and describe different techniques which allow us to determine for which short resistances a given resistive bridging fault may be detected. There, we will also briefly discuss the particularities of feedback bridging faults. In the same chapter, we present our experimental results on the occurrence of double errors induced by resistive bridging faults. Finally, we will prove that irrespective of technological parameters detection of certain classes of resistive bridging faults can be guaranteed.

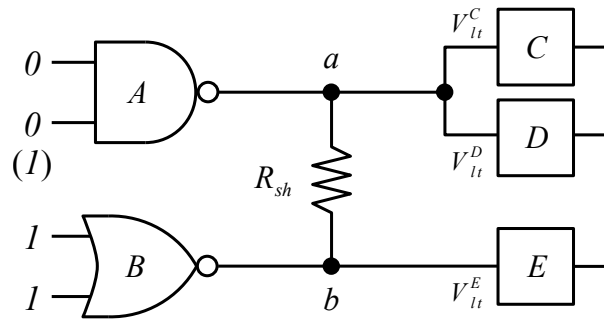


Figure 5.1: Example of circuit with a resistive short affecting nodes a and b .

5.1 Introduction of the Resistive Bridging Fault Model

In the following, we will introduce the findings of Renovell et al. [152, 154] as they have been covered in [W2, W3], [P4], and [J3]. Consider a resistive bridging fault affecting two nodes a and b driven by a two-input NAND gate A and a two-input NOR gate B , respectively (see Figure 5.1). Due to the input assignment $(0, 0)$ two parallel p -transistors in the pull-up network of gate A are active (refer to Figure 4.1 for the transistor structure of both gates). They drive node a to V_{dd} – corresponding to logical value 1 – in the fault-free case. Contrary to that, node b is driven to 0 V in the fault-free case – equaling the logical value 0 – by two parallel n -transistors in gate B . They are activated by assigning $(1, 1)$ to the gate’s inputs.

Yet, due to the short nodes a and b cannot attain their fault-free voltage level. Rather the voltages at both nodes are dependent on the defect’s intrinsic resistance. Figure 5.2(a) qualitatively illustrates the voltage $V_a(V_b)$ at node $a(b)$ on the ordinate as a function of the short’s resistance R_{sh} on the abscissa. As can be seen, V_a and V_b assume an equal intermediate voltage potential V_0 for $R_{sh} = 0\ \Omega$. In the presence of a short which has a non-zero intrinsic resistance, voltages in a and b monotonically approach the levels observable in the fault-free case. For an infinitely high short resistance both V_a and V_b reach the fault-free levels V_{dd} and 0 V , respectively. The characteristics of V_a and V_b have been verified by electrical simulations (see [154]).

In digital circuits the interpretation of voltage levels as Boolean values is of particular interest. As outlined in Chapter 2.1.2, this interpretation is done with respect to a single well-defined logic threshold voltage V_{it} which may be different for each input of each gate type. We assume – in accordance with previous works – that voltages can be interpreted without ambiguity. In their study of (non-resistive) bridging faults in an Advanced Micro Devices (AMD) design, Ma et al. [105] reported that disregarding potentially ambiguous intermediate voltages in the vicinity of the logic threshold had an impact on fault coverage which was below 0.007% . A possible variation of the logic threshold across different manufactured ICs is not considered. Furthermore, it is assumed that due to the high gain¹

¹Due to the high gain of CMOS logic gates, a voltage which slightly deviates from the logic threshold voltage of a gate’s input may be amplified to a very high (low) voltage at the gate’s output.

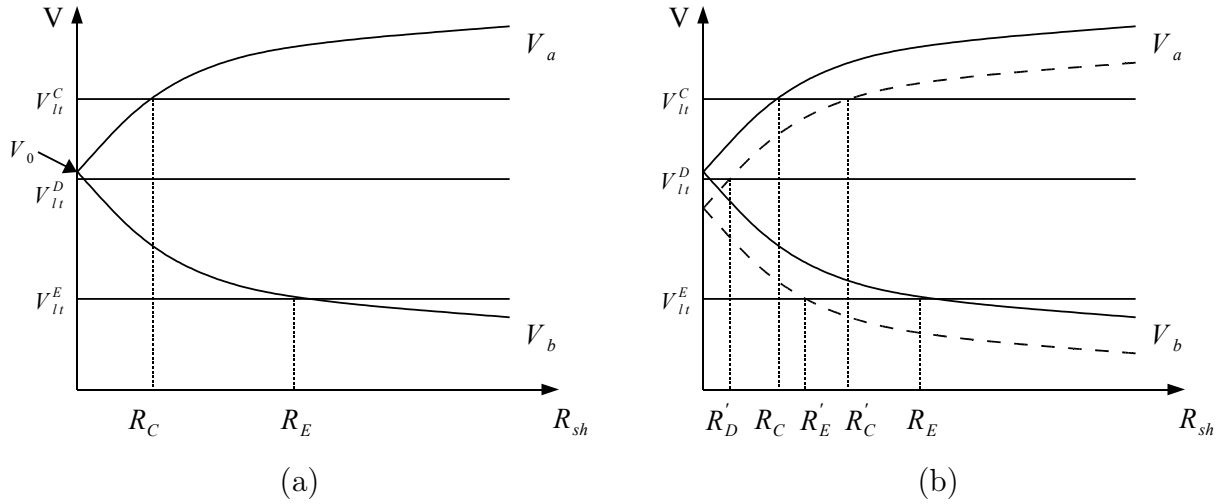


Figure 5.2: Characteristics of voltages at nodes a and b as a function of the short's resistance R_{sh} : (a) input assignment (0, 0, 1, 1), and (b) input assignments (0, 0, 1, 1) and (0, 1, 1, 1).

of CMOS logic (see e.g. [3, 146]), gates connected to the driven gates receive voltages corresponding to well-defined logical values at their inputs. Recently an extension to the resistive bridging fault model has been proposed by Cheung et al. [30]. It accounts for a range of uncertain interpretation surrounding each gate's logic threshold. Up to now, the necessity of this extension is not proven, as neither the typical extent of this uncertainty region nor its impact on the quality of results is sufficiently quantified in that work.

Nodes a and b in Figure 5.1 are observed by gates C , D and E , respectively. Note that in our example the types of these gates as well as the logic state of any non-defect affected inputs are irrelevant. In Figure 5.2(a) the logic threshold V_{lt}^C of gate C 's input – driven by node a – is depicted as a horizontal line. The resistance obtained when projecting the intersection between characteristic V_a and the threshold of C to the abscissa is called the *critical resistance*, denoted by R_C in this case. For shorts having a resistance smaller than R_C , the voltage characteristic of node a is below V_{lt}^C , i.e. the voltage would be interpreted as the faulty logical value 0 by C . The characteristic V_a is above the threshold for any short having a resistance greater than R_C . Consequently, in presence of such a defect, gate C would interpret the fault-free logical value 1. Obviously, only for shorts with an intrinsic resistance smaller than R_C , a faulty logical value would be interpreted by C . In this sense R_C constitutes an upper bound on the resistance of shorts. The conducting path created by these low ohmic defects would have a very strong influence on the voltage potentials of the shorted nodes and would be able to pull the voltage at a below V_{lt}^C . Inversely, for R_{sh} greater than R_C the short's influence would be too little to sufficiently affect V_a – gate C would interpret logical value 1.

Gate E in our example is driven by node b . The intersection of gate E 's logic threshold V_{lt}^E and V_b defines the critical resistance R_E . In the presence of a short having a resistance smaller(greater) than R_E , gate E would interpret the faulty(fault-free) logical value 1(0). Since there is no intersection between V_a and V_{lt}^D , there is no critical resistance for gate D .

As a consequence, this gate would interpret a fault-free logical value 1 irrespective of the short's resistance.

Assume that we now apply the pattern $(0, 1, 1, 1)$ to the driving gates A and B . In contrast to the first pattern $(0, 0, 1, 1)$, this one only activates one p -transistor in the pull-up network of A , the assignment to gate B and thus the strength of the gate's pull-down network remains unchanged. Because of the weakened pull-up network both voltage characteristics may be shifted towards $0V$, as illustrated by the dashed curves in Figure 5.2(b). Note that the characteristics depicted by continuous lines in this figure are the ones obtained for the first pattern. Clearly the logic thresholds of C , D , and E are invariant to the changed input assignment. Due to the shifted curves, however, we get new intersections and thus new critical resistances R'_C and R'_E for gates C and E , respectively. Remarkably for this pattern there is also an intersection between V_a and V_{it}^D , the logic threshold of gate D , resulting in the critical resistance R'_D .

We can observe that the maximum range of short resistances for which faulty logical values are interpreted by any driven gate differs for the two patterns. When the first pattern $(0, 0, 1, 1)$ is applied faulty logical values are observed for $0\Omega \leq R_{sh} < R_E$, while for the second one $(0, 1, 1, 1)$ this range is $0\Omega \leq R_{sh} < R'_C$. Since in Figure 5.2(b) it holds that $R'_C < R_E$ it is easy to see that the first pattern is able to expose defects from a larger resistance range. Hence, in this sense the first pattern is superior to the second one.

From our observations we conclude that based on the critical resistance we can identify those shorts which induce faulty logical values in a circuit. Moreover, the critical resistance R_{crit} which is associated with the input i of a gate driven by the shorted node n is affected by two factors:

1. the fixed logic threshold V_{it}^i of the driven gate's input i and
2. the voltage characteristic V_n of the shorted node n .

The voltage characteristic is a function of the short's resistance $R_{sh} \in \mathbb{R}_{\geq 0}$. Furthermore, it is dependent on the electrical properties of the active transistor networks in the driven gates and on the networks' topologies. The active transistors in turn are determined by the assignment to the inputs of the driven gates. Consequently the formation of fault effects caused by resistive bridging faults is pattern-dependent. The voltages at the shorted nodes are interpreted with respect to the individual logic thresholds of each driven gate. This means that the resistive bridging model also captures the Byzantine General's Problem.²

For a fixed input assignment and driven gate input the critical resistance $R_{crit} \in \mathbb{R}_{>0}$ is defined as the resistance for which $V_n(R_{crit}) = V_{it}^i$. Note that if no such R_{crit} exists this implies that the respective gate does not interpret a faulty logical value for any short under

²Note that both voting and biased voting model (discussed in Chapters 4.2.1 and 4.2.2, respectively), implicitly determine V_0 . In particular, the biased voting model derives the same logical values as those observed for $0 \leq R_{sh} < R_{min}$ (with R_{min} being the minimum critical resistance obtained for the current pattern).

the given input assignment. Based on R_{crit} we can identify two sets of short resistances:

$$F = \{x \in \mathbb{R}_{\geq 0} \mid 0 \leq x < R_{\text{crit}}\} \quad (5.1.1)$$

$$G = \{x \in \mathbb{R}_{> 0} \mid R_{\text{crit}} < x < \infty\} \quad (5.1.2)$$

Let $v \in \mathbb{B}$ be the logical value driven at node n in the fault-free case. If $R_{\text{sh}} \in F$ holds, input i will interpret a faulty logical value \bar{v} . For $R_{\text{sh}} \in G$, however, i will interpret the logical value v observed in the fault-free circuit. The RBF model cannot unambiguously resolve the logical value for $R_{\text{sh}} = R_{\text{crit}}$. Yet, for the fault coverage metrics introduced in Chapter 5.4, which are defined for integrals of resistance ranges, this can be neglected. Values for finitely many single points do not contribute to the improper integral of a monotonic function for (half-)open intervals. In the following we will denote this relation between short resistance and logical value by the interval $[0, R_{\text{crit}}] \bar{v}/v$.³ Alternatively we could describe the same situation by $[R_{\text{crit}}, \infty] v/\bar{v}$. Note that this notation does not discriminate between faulty and fault-free logical value.

To return to the example discussed above: When applying pattern $(0, 0, 1, 1)$ to the driven gates, gate C interprets logical value 0 for $R_{\text{sh}} < R_C$ and logical value 1 for $R_{\text{sh}} > R_C$. This can be indicated by $[0, R_C] 0/1$ and $[R_C, \infty] 1/0$, respectively.

5.2 Calculating Critical Resistances

As we have already pointed out in Chapter 5.1, critical resistances are technology dependent. A procedure which derives the critical resistance for the input of a driven gate has to take into account the input's logic threshold and the characteristics of the voltages at the shorted nodes. Both factors are influenced by the electrical properties of the circuit. Two conceptually different approaches for such a procedure are known. The first technique – favored in [30, 98, 165, 166] – performs an electrical simulation of both driven gates and the driving gate for which a critical resistance is to be determined. To avoid repetition of time consuming simulation runs the computed critical resistances are stored in look-up-tables. Consequently all possible combinations of gates from a gate library only have to be simulated and stored once for a certain circuit technology.

Unfortunately, electrical simulations have to be repeated in case the technology changes, even if only some of the parameters are affected. Therefore we prefer the second, more flexible approach, proposed by Huc [72, pp. 72] and Renovell et al. [154]. It is based on a general framework of electrical equations which exploit the functions describing the output characteristics (I_{ds} - V_{ds} characteristics) of each transistor and is thus very flexible. In the following we will first introduce the general framework in Chapter 5.2.1. Then Chapter 5.2.2 will reproduce how this framework was used by Huc and Renovell et al. to enable the calculation of critical resistances. This description has been adapted from our publication [J3]. Finally, in the same chapter we will introduce our instantiations of the electrical framework for two recent technology nodes which have been reported in [P8].

³Sar-Dessai et al. refer to this range as *detectable resistance interval* in [166].

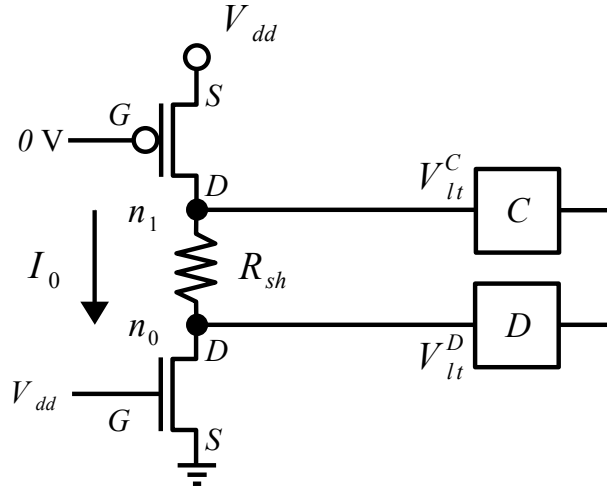


Figure 5.3: Transistor level view of a resistive bridging fault shorting two inverters.

5.2.1 General Framework

Figure 5.3 depicts a bridging fault shorting the outputs of two inverters (refer again to Figure 2.4 for the transistor structure of an inverter gate). The fault is activated as logical value 0, equaling 0 V, is applied to the inverter driving node n_1 . Thus, the p -transistor of this gate creates a connection between V_{dd} and the gate's output. Furthermore, logical value 1 which corresponds to V_{dd} , is applied to the inverter driving node n_0 . Consequently the n -transistor of this gate connects the ground terminal to the gate's output. Note that the inactive n -transistor(p -transistor) network in the upper(lower) gate is omitted in the figure. Node $n_1(n_0)$ is observed by gate $C(D)$ with the logic threshold $V_{lt}^C(V_{lt}^D)$. The short introduces a conducting connection with resistance R_{sh} between nodes n_1 and n_0 . As a consequence there is a current path linking the positive and negative power supply terminals, V_{dd} and V_{ss} (or ground potential), respectively. This path encompasses the p -transistor, the defect, and the n -transistor. No other current sources or sinks exist on this path. Hence, the same current is flowing through the three elements – we will denote it by I_0 . Current I_0 , as well as voltage V_{n_1} at node n_1 and voltage V_{n_0} at node n_0 are dependent on R_{sh} .

The current flowing between D and S terminal of a transistor is specified by the transistor's I_{ds} - V_{ds} characteristic. Naming conventions for the difference in voltage potentials between the transistor terminals are listed in Table 5.1. Note that the bulk terminal – which we omit in all figures – is commonly held at $V_{dd}(V_{ss})$ for a p -transistor(n -transistor). In this chapter we will assume $V_{dd} > V_{ss} = 0$ V. Let $I_p(V_{ds})$ and $I_n(V_{ds})$ be the output characteristic of the p - and n -transistor, respectively. Then for the situation illustrated in Figure 5.3 the following system of equations has to hold (see [72, p. 74] and [100]):

$$\begin{aligned}
 I_0 &= I_p(V_{dd} - V_{n_1}) \\
 I_0 &= I_n(V_{n_0}) \\
 R_{sh} &= (V_{n_1} - V_{n_0})/I_0
 \end{aligned} \tag{5.2.1}$$

Table 5.1: Naming conventions for the difference in the voltage potentials between the transistor terminals.

Symbol		Difference in voltage potential between
n -transistor	p -transistor	
$V_{ds,n}$	$V_{ds,p}$	Drain and source
$V_{gs,n}$	$V_{gs,p}$	Gate and source
$V_{bs,n}$	$V_{bs,p}$	Bulk and source

Suppose we want to determine the critical resistance R_C for the input of gate C (which is driven to logical value 1 in a fault-free circuit) from Figure 5.3. This means we have to determine R_{sh} in the system of equations for which $V_{n_1} = V_{it}^C$ (refer again to Chapter 5.1). Assume that the inverse functions of I_n and I_p are available and denoted by I_n^{-1} and I_p^{-1} , respectively. Since we know I_p and V_{dd} we can easily compute $I_0 = I_p(V_{dd} - V_{n_1})$. Next we obtain V_{n_0} using the inverse function I_n^{-1} of I_n as $V_{n_0} = I_n^{-1}(I_0)$. Finally we can calculate the critical resistance solving $R_C = R_{sh} = (V_{n_1} - V_{n_0})/I_0$. To determine the critical resistance R_D for the input of gate D (which is driven to logical value 0) we set $V_{n_0} = V_{it}^D$ and calculate $I_0 = I_n(V_{n_0})$. Using the inverse function I_p^{-1} of I_p we obtain V_{n_1} and finally $R_D = R_{sh}$.

This framework reduces critical resistance calculation to the problem of solving a simple system of equations. At the same time it allows for a high degree of flexibility. Changes affecting the technological parameters can now be easily addressed. Furthermore the framework is applicable to any arbitrary CMOS technology (including company-internal solutions) provided that I_{ds} - V_{ds} characteristics for p - and n -transistors and their inverse functions are known. Output characteristics are a common means to describe the behavior of transistors and are thus widely available. In the following we will demonstrate how to instantiate the general framework for a particular technology's I_{ds} - V_{ds} characteristics to obtain a *technology specific model*. To underline the framework's flexibility we will discuss instantiations for three different technology generations. Experimental results we performed to compare the three models with respect to their influence on test pattern generation and fault simulation can be found in Chapter 8.2.2.

5.2.2 Technology-Specific Models

The first model, proposed by Huc and Renovell et al., uses the Shockley equations from the 1940s [175] and is valid for conventional technologies. We will refer to it as *Shockley model*. More recent technologies, for which Shockley's equations are not necessarily valid, can be addressed by our *Fitted model*. We have obtained this model by fitting data from electrical simulations. For current deep sub-micron manufacturing technologies our *Predictive model* can be used. It is based on Berkeley Predictive Technology Model [22] (BPTM), which is provided by the Device Group at UC Berkeley, in combination with Berkeley Short-channel

Table 5.2: Technological parameters for Shockley and Fitted model (n - and p -channel MOS-FET).

Symbol		Meaning
n -transistor	p -transistor	
C_{ox}	C_{ox}	oxide capacitance per area unit
μ_n	μ_p	mobility
W_n	W_p	channel width
L_n	L_p	channel length
V_{tn0}	V_{tp0}	zero bias threshold voltage
Γ_n	Γ_p	body effect coefficient
Φ_n	Φ_p	substrate potential

IGFET Model 4 (BSIM4), which is valid for 90 nm technologies.⁴ In particular, it can be used for predicting the transistor behavior in future 65 nm and 45 nm technologies. A new iteration of the model reported in [214] can even predict transistor behavior down to 32 nm technology. BPTM/BSIM4 accounts for the numerous non-trivial electrical phenomena in nanoscale technologies, including Non-Uniform Lateral Doping (NULD), Narrow Width Effect, Short-Channel Effect, Drain-Induced Barrier Lowering (DIBL), Drain-Induced Threshold Shift (DITS) and Bulk Charge Effect. Hence, the three models describe past, present and future technologies; nevertheless they share the same basic framework.

Shockley Model

For the linear operation region of a n -transistor Shockley's equation for the drain current $I_{ds,n}$ as a function of the drain-to-source voltage $V_{ds,n}$ is given as:

$$I_{ds,n}(V_{ds,n}) = \mu_n C_{ox} \frac{W_n}{L_n} \left((V_{gs,n} - V_{tn0}) V_{ds,n} - \frac{V_{ds,n}^2}{2} \right). \quad (5.2.2)$$

Refer to Table 5.2 for an explanation of the symbols used.

Similarly, for a p -transistor the drain current $I_{ds,p}$ as a function of the drain-to-source voltage $V_{ds,p}$ is governed by:

$$I_{ds,p}(V_{ds,p}) = -\mu_p C_{ox} \frac{W_p}{L_p} \left((V_{gs,p} - V_{tp0}) V_{ds,p} - \frac{V_{ds,p}^2}{2} \right). \quad (5.2.3)$$

Observing that the n -transistor's source terminal is connected to ground potential and that logical value 1, equaling to V_{dd} , is applied to its gate, $V_{gs,n} = V_{dd}$ has to hold (refer to

⁴When our paper [P8] was published, BSIM4.4.0 [208] was available. Our discussion, being based on [P8], is thus restricted to this version of BSIM4. The updated version BSIM4.6.1, available since May 2007, might induce changes to our material presented in the following.

Figure 5.3). Furthermore, $V_{ds,n}$ equals V_{n0} . Hence, Equation (5.2.2) can be transformed to obtain the current flowing through the n -transistor as a function of the voltage at node n_0 :

$$I_n(V_{n0}) = \mu_n C_{ox} \frac{W_n}{L_n} \left((V_{dd} - V_{tn0})V_{n0} - \frac{V_{n0}^2}{2} \right). \quad (5.2.4)$$

As logical value 0, equaling 0 V, is applied to the gate of the p -transistor and its source terminal is connected to V_{dd} it holds that $V_{gs,p} = -V_{dd}$ and $V_{ds,p} = (V_{n1} - V_{dd})$. Furthermore for a p -transistor $V_{tp0} < 0$ is true. Now Equation (5.2.3) can be transformed, taking into account that $I_0 = -I_{ds,p}$ and by substituting $V_{ds,p}$ and $V_{gs,p}$. The resulting equation describes the current flowing through the p -transistor as a function of the voltage at the node n_1 :

$$I_p(V_{n1}) = \mu_p C_{ox} \frac{W_p}{L_p} \left((V_{dd} - |V_{tp0}|)(V_{dd} - V_{n1}) - \frac{(V_{dd} - V_{n1})^2}{2} \right). \quad (5.2.5)$$

We need to know the logic threshold V_{it} of the gate's input, in order to calculate the critical resistance for a gate connected to the shorted node driven to logical value 0 according to the procedure described in Chapter 5.2.1. Furthermore, I_n from Equation (5.2.4) and I_p^{-1} are required. The inverse equation of I_p can be obtained by solving the quadratic equation $I_0 = I_p(V_{n1})$ (Equation (5.2.5)). In [72, pp. 74], [154], and [J3] the following closed-form formula for the critical resistance of a gate that is driven by the n -transistor network has been proposed:

$$R_{\text{crit},n} = \frac{|V_{tp0}| - V_{it} + \sqrt{(V_{dd} - |V_{tp0}|)^2 - \frac{2I_n(V_{it})}{C_{ox}\mu_p W_p/L_p}}}{I_n(V_{it})}. \quad (5.2.6)$$

Similarly for a gate connected to the shorted node driven to logical value 1 we require the gate's logic threshold V_{it} , I_p from Equation (5.2.5), and I_n^{-1} . The latter can be obtained from Equation (5.2.4). The closed-form formula for the critical resistance of a gate that is driven by the p -transistor network is (see [72, pp. 74], [154], and [J3]):

$$R_{\text{crit},p} = \frac{V_{it} - V_{dd} + V_{tn0} + \sqrt{(V_{dd} - V_{tn0})^2 - \frac{2I_p(V_{it})}{C_{ox}\mu_n W_n/L_n}}}{I_p(V_{it})}. \quad (5.2.7)$$

If in both driving gates only a single transistor is active (as in the inverter example in Figure 5.3) both Equations (5.2.6) and (5.2.7) can be used directly. The transistors' channel widths and lengths simply have to be inserted into the equation. For gates with more complex pull-up and pull-down networks the width and length of all active transistors first has to be combined to form a single *equivalent transistor*. The set of active transistors for each driving gate is determined by the gate's input assignment.

For active transistors connected in parallel (such as e.g. the p -transistors in a NAND gate) this simply means summing up the width/length ratio of each individual transistor. Let

the width/length ratio of the i -th n -transistor(p -transistor) be $(W_n/L_n)_i((W_p/L_p)_i)$. Then the width/length ratio $(W/L)_{\text{equ},n}$ of the equivalent transistor for n -channel networks and the ratio $(W/L)_{\text{equ},p}$ for p -channel networks, respectively, are computed as follows:

$$\left(\frac{W}{L}\right)_{\text{equ},n} = \sum_{i=1}^k \left(\frac{W_n}{L_n}\right)_i \quad (5.2.8)$$

$$\left(\frac{W}{L}\right)_{\text{equ},p} = \sum_{i=1}^k \left(\frac{W_p}{L_p}\right)_i \quad (5.2.9)$$

The resulting width/length ratio of the equivalent transistor can now be used in Equations (5.2.6) and (5.2.7) – just as in the single transistor case.

If the active transistors are connected in series (such as e.g. the n -transistors in a NAND gate) this process is more complex. One solution which takes body-bias effect and substrate potential into account has been reported in [72, pp. 47] and [155]. The width/length ratio $(W/L)_{\text{equ},n}$ of the equivalent transistor for a network of k n -channel transistors is given by:

$$\left(\frac{W}{L}\right)_{\text{equ},n} = \frac{1}{\sum_{i=1}^k \left(\frac{L_n}{W_n}\right)_i} \cdot Cor_n(k, V_0) \quad (5.2.10)$$

$$Cor_n(k, V_0) = 1 - \frac{\Gamma_n \left(\sqrt{\Phi_n + V_0 \frac{k-1}{2k}} - \sqrt{\Phi_n} \right)}{V_{\text{dd}} - V_{\text{tn}0} - \frac{V_0}{2}}$$

For a network of k p -channel transistors the width/length ratio $(W/L)_{\text{equ},p}$ of the equivalent transistor is calculated as:

$$\left(\frac{W}{L}\right)_{\text{equ},p} = \frac{1}{\sum_{i=1}^k \left(\frac{L_p}{W_p}\right)_i} \cdot Cor_p(k, V_0) \quad (5.2.11)$$

$$Cor_p(k, V_0) = 1 - \frac{\Gamma_p \left(\sqrt{\Phi_p + (V_{\text{dd}} - V_0) \frac{k-1}{2k}} - \sqrt{\Phi_p} \right)}{V_{\text{dd}} - |V_{\text{tp}0}| - \frac{V_{\text{dd}} - V_0}{2}}$$

As Huc could demonstrate in [72, pp. 53] for $V_0 = V_{\text{dd}}$ the accuracy of Equations (5.2.10) and (5.2.11) is sufficiently close to the results obtained from electrical simulations. Increased correlation can be achieved by setting $V_0 = V_{\text{lt}}$, where V_{lt} is the logic threshold voltage of the gate input, for which the critical resistance is to be calculated.

For some complex gate types, pull-up and pull-down network consist of combinations of parallel and serial transistors. These cases can be resolved by iteratively applying Equations (5.2.10), (5.2.8), and (5.2.11), (5.2.9), respectively (for details see e.g. [72, p. 55]).

Fitted Model

For our Fitted model⁵ the n -transistor's output characteristic is described by:

$$I_n(V_{n0}) = A_n W_n \left((V_{dd} - B_n) V_{n0} - \frac{V_{n0}^2}{2} \right). \quad (5.2.12)$$

The corresponding equation for a p -transistor is:

$$I_p(V_{n1}) = A_p W_p \left((V_{dd} - |B_p|)(V_{dd} - V_{n1}) - \frac{(V_{dd} - V_{n1})^2}{2} \right). \quad (5.2.13)$$

We obtained parameters A_n , A_p , B_n , and B_p used in both Equation (5.2.12) and (5.2.13) by fitting results of electrical simulations. These parameters are specific for each transistor configuration of each gate type. In particular, they also account for the impact of multiple transistors being active within one network. Hence, calculation of equivalent transistors – as in the Shockley model – is not required. Note that for $A_n = \mu_n C_{ox} / L_n$, $A_p = \mu_p C_{ox} / L_p$, $B_n = V_{tn0}$, and $B_p = V_{tp0}$ Equations (5.2.12) and (5.2.13) equal their respective counterparts for the Shockley model. We also experimented with fitting results of electrical simulations using the Alpha-Power Law model [164]. Yet, we were able to obtain more accurate results using Equations (5.2.12) and (5.2.13).

Based on I_n from Equation (5.2.12), we compute the critical resistance for a gate connected to the shorted node driven to logical value 0 (i.e. by an n -transistor) as follows:

$$R_{\text{crit},n} = \frac{|B_p| - V_{\text{it}} + \sqrt{(V_{dd} - |B_p|)^2 - \frac{2I_n(V_{\text{it}})}{A_p W_p}}}{I_n(V_{\text{it}})}. \quad (5.2.14)$$

For a gate connected to the shorted node driven to logical value 1 by a p -transistor the corresponding equation is (where I_p according to Equation (5.2.13) is used):

$$R_{\text{crit},p} = \frac{V_{\text{it}} - V_{dd} + B_n + \sqrt{(V_{dd} - B_n)^2 - \frac{2I_p(V_{\text{it}})}{A_n W_n}}}{I_p(V_{\text{it}})}. \quad (5.2.15)$$

To evaluate the accuracy of the Shockley and our Fitted model we performed a comparison with simulation data obtained from the electrical simulator HSPICE. We took parameters from the model card of a 0.35 μm technology from austriamicrosystems AG (AMS), Austria. Results for a bridging fault shorting the outputs of two inverters (columns 2 to 4), and a fault affecting the outputs of two NAND gates can be found in Table 5.3. For the latter case we distinguish between one and two active p -transistors in the NAND gate's pull-up network (in columns 5 to 7 and 8 to 10, respectively). Furthermore we compared several different widths for both n - and p -transistors. Note that for the bridge between two inverters, critical resistances were computed for a gate driven by the p -transistor. For

⁵We are thankful to Prof. P. Maurine (LIRMM, France) for his insights on the fitting approach.

Table 5.3: Results of Shockley and Fitted models compared to HSPICE; $L_p = L_n = 0.35 \mu\text{m}$, $V_{\text{dd}} = 3.3 \text{V}$, logic threshold $V_{\text{it}} = V_{\text{dd}}/2$.

	2-inverter bridge			2-NAND bridge, 1 active p -transistor			2-NAND bridge, 2 active p -transistors		
W_p [μm]	2	4	8	2	4	8	2	4	8
W_n [μm]	1	2	4	1	2	4	1	2	4
	$R_{\text{crit},C}$ [Ω]			$R_{\text{crit},D}$ [Ω]			$R_{\text{crit},D}$ [Ω]		
HSPICE	2,422	1,159	567	1,686	838	418	4,250	2,064	1,018
Shockley	2,304	1,152	576	2,305	1,152	576	4,660	2,330	1,165
Fitted	2,415	1,157	566	1,685	838	419	4,266	2,072	1,019

the two-NAND case, resistances computed for the gate driven by the n -transistor network are stated.

From our data it is evident that the deviation between the critical resistances obtained using our Fitted model and the HSPICE simulated values is very low – in fact it never exceeds 0.4%. In contrast to that, the comparison of the resistances computed with the Shockley model and the simulated data yields a much larger discrepancy. For the – rather simple – two inverter case the deviation is already up to 5%. In the two-NAND case, even differences larger than 35% could be observed.

Predictive Model

For BPTM/BSIM4⁶ the output characteristic of an n -channel transistor in the relevant $V_{ds,n}$ region is given by:

$$I_{ds,n}(V_{ds,n}) = \frac{\frac{W_n}{L_n} \mu_{\text{eff},n} Q_{ch0,n} V_{ds,n} \left(1 - \frac{V_{ds,n}}{2V_{b,n}}\right)}{1 + \frac{V_{ds,n}}{E_{\text{sat},n} \cdot L_n}}. \quad (5.2.16)$$

where $\mu_{\text{eff},n}$ is the effective mobility, $Q_{ch0,n}$ is the channel charge density and $E_{\text{sat},n}$ is the critical electrical field at which the carrier velocity becomes saturated. $V_{b,n}$ is defined as $(V_{g\text{steff},n} + 2v_t)/A_{\text{bulk},n}$, where $V_{g\text{steff},n}$ is the effective $(V_{gs,n} - V_{th,n})$, v_t is the thermal voltage ($k_B T/q$) and A_{bulk} models the bulk charge effect. All these parameters can be calculated from over 100 technology dependent process parameter values. Parameter sets for technologies ranging from 130 nm down to 32 nm are provided by the authors of [22, 214].

The equation $I_{ds,p}(V_{ds,p})$ for the output characteristic of p -transistors is essentially the same as Equation (5.2.16). However, parameters $\mu_{\text{eff},n}$, $Q_{ch0,n}$ etc. have to be replaced by their p -transistor equivalents (indicated by subscript “ p ”).

⁶We are thankful to Prof. B. Nikolic (UC Berkeley, USA) for his hint on BPTM/BSIM4.

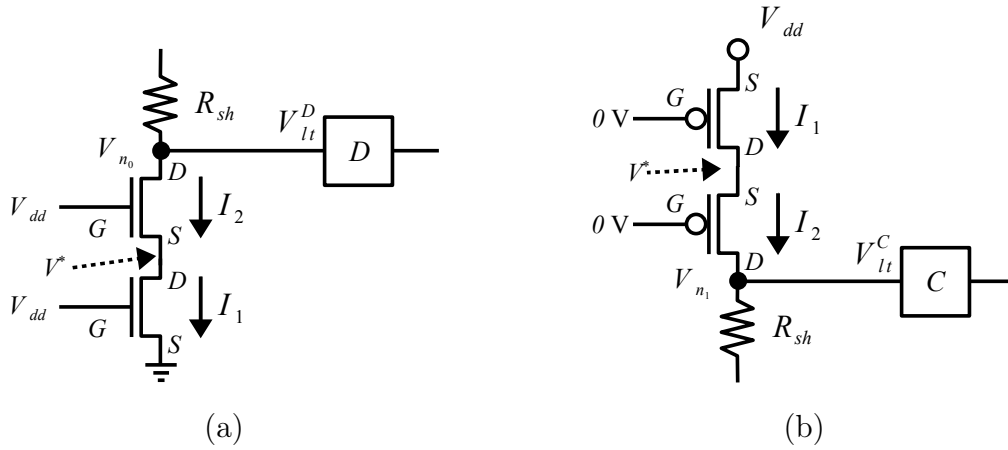


Figure 5.4: Transistor level view of a series network with (a) two n -transistors, and (b) two p -transistors.

From Equation (5.2.16) and its p -transistor equivalent, we can derive I_n and I_p in a straightforward manner. To compute I_0 from V_{n_0} for a single n -transistor using I_n , we have to set $V_{bs,n}$ to 0 V and $V_{gs,n}$ to V_{dd} . Given I_0 , V_{n_1} can be computed for a single p -transistor using I_p and again setting $V_{bs,p}$ to 0 V and $V_{gs,p}$ to $-V_{dd}$.

Algorithm 5.1: Computation of $V_{n_0} = I_n^{-1}(I_0)$ for a single (or parallel) n -transistor(s).

Input: Current I_0

Output: Voltage V_{n_0}

```

/* Calculate  $V' = V_{n_0}$  assuming  $V_{ds,n} = 0$  V */
1 Extract parameters for  $V_{ds,n} = 0$  V;
2 Calculate  $V' := I_n^{-1}(I_0)$  for  $V_{bs,n} = 0$  V and  $V_{gs,n} = V_{dd}$ ;
/* Repeat lines 1 and 2 for  $V_{ds,n} = V'$  */
3 Extract parameters for  $V_{ds,n} = V'$ ;
4 Calculate  $V' := I_n^{-1}(I_0)$  for  $V_{bs,n} = 0$  V and  $V_{gs,n} = V_{dd}$ ;
5 return  $V'$ ; /* return  $V_{n_0}$  */
    
```

The inverse equations I_n^{-1} and I_p^{-1} are obtained – just as for Shockley and Fitted model – by solving a quadratic equation. Unfortunately, parameters $V_{gsteff,n}$, $\mu_{eff,n}$, $Q_{ch0,n}$, and $V_{b,n}$ are (implicitly) dependent on the transistor threshold, which in turn depends on $V_{ds,n}$. Obviously when calculating $V_{ds,n} = I_n^{-1}(I_0)$, $V_{ds,n}$ is unknown beforehand. The same interdependency problem exists for p -transistors as well.

To solve this problem for a single n -transistor we propose Algorithm 5.1. Given I_0 it calculates V_{n_0} in a two step process. First we assume $V_{ds,n} = 0$ V (for p -transistors: $V_{ds,p} = 0$ V) and extract all parameters relevant for solving I_n^{-1} . Then, in line 2, we determine the voltage V' at node n_0 under this assumption (for p -transistors the voltage at n_1 is calculated using I_p^{-1} with $V_{bs,p} = 0$ V and $V_{gs,p} = -V_{dd}$). Next we re-extract the parameters assuming $V_{ds,n} = V'$ (for p -transistors: $V_{ds,p} = V'$) and repeat the calculation

of V' (line 4). The resulting voltage $V_{n_0} = V'$ is returned in line 5 (for p -transistors $V_{n_1} = (V_{\text{dd}} + V')$ is returned). We have found that the maximum deviation between the two consecutive computations of V' never exceeded 2 mV and that changes in current were below $10^{-14} \mu\text{A}$. When optimizing Algorithm 5.1 for speed, the second iteration can thus be omitted without significant loss of accuracy.

As for the width/length ratio of parallel n - and p -transistors, the same additive relations expressed in Equations (5.2.8) and (5.2.9) hold. Hence, I_n , I_p and their inverse counterparts can also be used for parallel networks of active transistors.

Again, the handling of serial transistor networks is more complicated. Consider the case of two n - and two p -transistors connected in series as depicted in Figure 5.4(a) and (b), respectively. The complementary active network in the second driven gate is omitted in both figures. By V^* we will denote the voltage potential attained “in between” the two serial transistors. Symbol I_1 denotes the current flowing through the transistor which is directly connected to V_{dd} and ground terminal, respectively. The current flowing through the second transistor in the network (connected to the gate’s output) is referred to as I_2 .

Algorithm 5.2: Computation of $I_0 = I_n(V_{n_0})$ for two n -transistor in series.

Input: Voltage V_{n_0} , threshold ε

Output: Current I_0

```

1  $V^* := V_{n_0}/2$ ;
2 while (true) do
   | /* Determine  $I_1$  and  $I_2$  using Equation (5.2.16) */
3   | Calculate  $I_1 := I_n(V^*)$  with  $V_{bs,n} = 0\text{ V}$  and  $V_{gs,n} = V_{\text{dd}}$ ;
4   | Calculate  $I_2 := I_n(V_{n_0} - V^*)$  with  $V_{bs,n} = -V^*$  and  $V_{gs,n} = V_{\text{dd}} - V^*$ ;
5   |  $I^* := (I_1 + I_2)/2$ ;
6   | if ( $|I_1 - I_2| < \varepsilon$ ) then return  $I^*$ ; /* return  $I_0$  */
   | /* Calculate  $V^*$  from  $I^*$  */
7   | Extract parameters for  $V_{ds,n} = 0\text{ V}$ ;
8   | Calculate  $V^* := I_n^{-1}(I^*)$  for  $V_{bs,n} = 0\text{ V}$  and  $V_{gs,n} = V_{\text{dd}}$ ;
9 end

```

The current flowing through both transistors has to be the same, i.e. $I_1 = I_2$ has to hold. This property is exploited by our Algorithm 5.2 which, given V_{n_0} , computes $I_0 = I_n(V_{n_0})$ for two n -transistors connected in series. Our algorithm approximates the true value of (the initially unknown voltage) V^* until the absolute difference between I_1 and I_2 is smaller than a predefined value ε . The average of those two currents is an approximation of I_0 .

In the beginning, $V^* = V_{n_0}/2$ is assumed in line 1 (for p -transistors we presume $V^* = (V_{n_1} + V_{\text{dd}})/2$). Within the loop, first I_1 and I_2 are calculated in lines 3 and 4, respectively. (For p -transistors we compute $I_1 = I_p(V^* - V_{\text{dd}})$ for $V_{bs,p} = 0\text{ V}$ and $V_{gs,p} = -V_{\text{dd}}$ in line 3, and $I_2 = I_p(V_{n_1} - V^*)$ for $V_{bs,p} = (V_{\text{dd}} - V^*)$ and $V_{gs,p} = -V^*$ in line 4.) Once the difference between both currents is below ε our algorithm exits by returning the approximated value of I_0 (line 6). Otherwise a new estimation for V^* is determined based on the average of

I_1 and I_2 in line 8 (for $V^* = V'$ this is equivalent to line 2 in our Algorithm 5.1) and a new iteration begins. For $\varepsilon = 1 \mu\text{A}$ we could observe that our algorithm always terminated after six iterations.

Algorithm 5.3: Computation of $V_{n_0} = I_n^{-1}(I_0)$ for two n -transistors in series.

Input: Current I_0

Output: Voltage V_{n_0}

```

/* Calculate  $V^*$  and  $V'$  assuming  $V_{ds,n} = 0 \text{ V}$  */
1 Extract parameters for  $V_{ds,n} = 0 \text{ V}$ ;
2 Calculate  $V^* = I_n^{-1}(I_0)$  for  $V_{bs,n} = 0 \text{ V}$  and  $V_{gs,n} = V_{dd}$ ;
3 Calculate  $V' = I_n^{-1}(I_0)$  for  $V_{bs,n} = -V^*$  and  $V_{gs,n} = V_{dd} - V^*$ ;
  /* Repeat lines 1-3 for  $V_{ds,n} = V^* + V'$  */
4 Extract parameters for  $V_{ds,n} = V^* + V'$ ;
5 Calculate  $V^* = I_n^{-1}(I_0)$  for  $V_{bs,n} = 0 \text{ V}$  and  $V_{gs,n} = V_{dd}$ ;
6 Calculate  $V' = I_n^{-1}(I_0)$  for  $V_{bs,n} = -V^*$  and  $V_{gs,n} = V_{dd} - V^*$ ;
7 return  $V^* + V'$ ; /* return  $V_{n_0}$  */

```

For the inverse scenario, i.e. calculate $V_{n_0} = I_n^{-1}(I_0)$ for two n -transistors in series, we propose Algorithm 5.3. Its basic structure is similar to that of our Algorithm 5.1: For the initial voltage calculation parameters are extracted assuming that $V_{ds,n}$ equals 0 V. The resulting voltages are used to refine the expected value of $V_{ds,n}$; parameters are extracted again, and the final value of V_{n_0} is computed.

Voltage V_{n_0} for transistors in series is composed of two parts. One is contributed by the difference in potential between the ground terminal and V^* which is computed in lines 2 and 5 (for p -transistors I_p^{-1} is used for $V_{bs,p} = 0 \text{ V}$ and $V_{gs,p} = -V_{dd}$). The second contribution V' comes from the difference in potential between V^* and the gate's output driving n_0 ; it is calculated in lines 3 and 6 (for p -transistors $V_{bs,p} = (V_{dd} - V^*)$ and $V_{gs,p} = -V^*$). The resulting voltage V_{n_0} is returned in line 7 ($V_{n_1} = (V_{dd} + V^* + V')$ is the return value for p -transistors).

5.3 Analogue Detectability Intervals

Using simulation techniques described in Chapter 5.5, the intervals of short resistances introduced in Chapter 5.1 can be propagated to the observable points of a circuit. For a given bridging fault and an assignment to the circuit's inputs these techniques yield for each observable point the range of short resistances for which the fault is detectable at that point. In accordance with [149, 151], we will call such a range *analogue detectability interval* (ADI).⁷ Typically an ADI is of the form $[R_1, R_2]$ with R_1 equaling to 0Ω . However, due to reconvergencies of fault effects for instance, non-contiguous ADIs are also possible

⁷In [98] the range of detectable short resistances is called *detection condition set* (DCS).

(see [149]). In general, if we assume an ADI consisting of n disjoint ranges, we can write the range of detectable resistances – analogously to Equation (5.1.1) – as:

$$\{x \in \mathbb{R} | R_1 < x < R_2\} \cup \{x \in \mathbb{R} | R_3 < x < R_4\} \cup \dots \cup \{x \in \mathbb{R} | R_{2n-1} < x < R_{2n}\}$$

Note that we require $R_{2i} < R_{2i+1}$ for $i \in [1, n-1]$. Using a notation similar to the one introduced together with Equations (5.1.1) and (5.1.2), we will denote this by $\{[R_1, R_2], [R_3, R_4], \dots [R_{2n-1}, R_{2n}]\}$. For a circuit with m observable points this definition can be extended as follows: Let t be the test vector applied to the inputs of the circuit. We denote the ADI propagated by t to the observable point j for a given bridging fault f as $\text{ADI}_j(f, t)$. The union of all ADIs of f covered by a given test set $T = \{t_1, t_2, \dots, t_k\}$ is defined as $C\text{-ADI}(f) = \cup_{i=1}^k \cup_{j=1}^m \text{ADI}_j(f, t_i)$. The union of all ADIs of f which can be obtained for an exhaustive test set, i.e. a test set which contains all 2^n input combinations possible for a combinational circuit with n inputs, T is referred to as $G\text{-ADI}$ (where G stands for ‘global’). While $C\text{-ADI}$ contains all short resistances for which the bridging fault f can be detected by the test set given, $G\text{-ADI}$ includes all the resistances for which the fault is detectable by any test set. In particular this means that $C\text{-ADI}(f) \subseteq G\text{-ADI}(f)$ holds. We will consider all shorts at bridging fault location f whose intrinsic resistance R_{sh} is not covered by $G\text{-ADI}(f)$, i.e. $R_{\text{sh}} \notin G\text{-ADI}(f)$, as being *redundant*.⁸ $G\text{-ADI}$ does not necessarily have to be calculated using exhaustive fault simulation. It can be obtained as a “byproduct” of ATPG for resistive bridging faults as well (see Chapter 8). In general, it is unlikely that an efficient algorithm calculating $G\text{-ADI}$ can be constructed. Since, as Polian has proven in [136, pp. 75], the existence of a polynomial-time algorithm for the computation of $G\text{-ADI}$ would imply $P = NP$.

5.4 Fault Coverage Metrics

Commonly, the quality of a test set with respect to a set of faults is quantified by a single number – the fault coverage. In “traditional” fault models, such as e.g. the stuck-at model, the fault’s detection status is binary: Either the fault is detected or not. Contrary to that in the RBF model, a fault’s detection status is given by the (potentially empty) ADI. In Chapter 5.1 we have demonstrated that different test vectors may lead to different detection ranges. Consequently, they may also result in different ADIs. Hence, in order for a fault coverage metric to fully account for the quality of a test set with respect to the RBF model this resistance information may not be neglected.

In [36] the concept of a probabilistic fault coverage metric was developed for intra-gate bridging faults (it is based on ideas from [42]). The metric evaluates the detection probability of a resistive intra-gate bridging fault by calculating the expected value of a random variable $d \in [0, 1]$, where $d = 1$ ($d = 0$) means that the fault is detected (not detected). Assuming that a given test set detects the intra-gate bridging fault in question

⁸This type of redundancy is called “analog redundancy” in [149, 150] as opposed to “digital redundancy” which means that $G\text{-ADI}(f) = \emptyset$ and thus at f no defect of any short resistance is detectable by voltage testing.

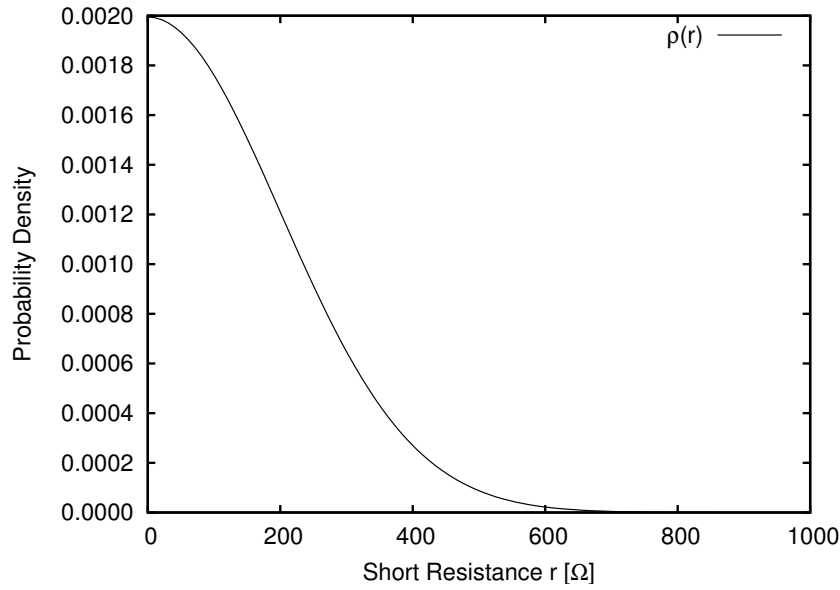


Figure 5.5: Probability density function ρ for short resistance r as proposed by Renovell et al. [154].

for all intrinsic resistances between 0Ω and R_T the expected value E of d is calculated as:

$$E(d) = \int_0^{R_T} \rho(r) dr. \quad (5.4.1)$$

In this equation ρ denotes the probability density function of the short's resistance, i.e. the probability that a short with a given resistance r occurs. In [154] Renovell et al. have obtained ρ by approximating data from an experimental study performed by Rodríguez-Montañés et al. [158] using a normal distribution. The experimental study reports that shorts with an intrinsic resistance between 0Ω and 500Ω are most common in CMOS. The probability density function derived according to the proposal by Renovell et al. is depicted in Figure 5.5. The authors of [165] use a slightly different approach to derive ρ for the same experimental data. A more recent work by Spica et al. [178] found that ρ is uniform for copper processes. In an industrial setting ρ may be determined using monitor structures manufactured in the same facility or even on the same wafer as the actual product.

The concept expressed in Equation (5.4.1) can be extended to ADIs and is thus applicable to resistive inter-gate bridging faults as well. Several probabilistic fault coverage metrics are known in literature. In the following we present a compilation of these metrics adapted from our publication [J3].⁹

⁹In the original papers these metrics are typically referred to as 'fault coverage' – to facilitate the discussion we will assign distinguishable names to them.

The *pessimistic fault coverage* P -FC proposed in [154] is defined for one fault f as:

$$P\text{-FC}(f) = 100\% \cdot \left(\int_{C\text{-ADI}(f)} \rho(r) dr \right) / \left(\int_0^{\infty} \rho(r) dr \right). \quad (5.4.2)$$

The range of resistances detected by the test set is “weighted” using $\rho(r)$. The numerator is then related to the “weighted” continuum of short resistances ranging from 0Ω to ∞ .¹⁰ It is very likely that even when considering any possible test set, a bridging fault cannot be detected for the whole continuum of short resistances (at least if effects on reliability, signal propagation delay and IDDQ are not taken into account). As already mentioned, we will regard shorts having a resistance from the remaining undetectable range as being redundant. Obviously, in the denominator of P -FC in Equation (5.4.2) redundant shorts are not excluded; this makes the metric pessimistic.

The fault coverage proposed in [166] focuses on the range of short resistances for which faulty logical values may be visible at the fault site. Let $R_{\max}(f)$ be defined as the maximum critical resistance which can be determined at fault location f for any activating assignment to the driving gates and any driven gate. For a fault location f , where the driving gates have n inputs, and there are m driven gates, determining the maximum critical resistance has a computational complexity of $O(2^n \cdot m)$. Usually both n and m are rather small, single digit numbers and only a subset of the 2^n assignments to the driving gates actually activates the bridging fault. As a consequence, the maximum critical resistance can be determined very quickly (using e.g. a procedure similar to Algorithm 5.4).

Once $C\text{-ADI}(f)$ and $R_{\max}(f)$ are known the *excitation based fault coverage* E -FC for bridging fault f is calculated as:

$$E\text{-FC}(f) = 100\% \cdot \left(\int_{C\text{-ADI}(f)} \rho(r) dr \right) / \left(\int_0^{R_{\max}(f)} \rho(r) dr \right). \quad (5.4.3)$$

By using $[0, R_{\max}(f)]$ instead of $[0, \infty]$ in the denominator, E -FC excludes all shorts which cannot be activated (i.e. excited) locally at the fault site. Yet, the metric does not consider whether fault effects can be propagated from the fault site to observable points. Additionally, it is neglected that there might be no assignment to the circuit’s inputs which justifies the logical values required to activate the bridge and to propagate faulty effects. Consequently, E -FC may still account for some redundant defects.

Based on G -ADI, [149] has proposed a metric we will refer to as *global fault coverage* G -FC. For a bridging fault f with non-empty $G\text{-ADI}(f)$ metric $G\text{-FC}(f)$ is defined as (in case $G\text{-ADI}(f) = \emptyset$, $G\text{-FC}(f)$ evaluates to 0%):

$$G\text{-FC}(f) = 100\% \cdot \left(\int_{C\text{-ADI}(f)} \rho(r) dr \right) / \left(\int_{G\text{-ADI}(f)} \rho(r) dr \right). \quad (5.4.4)$$

¹⁰Note that $\rho(r)$ is typically defined such that $\int_0^{\infty} \rho(r) dr$ approximates 1.

Since G -FC focuses on those resistances for which the fault is detectable it can be considered as the exact metric. Thus, in case $G\text{-FC}(f) < 100\%$ is yielded for a test set there is a high probability that defective ICs may pass that particular test. Unfortunately, as the calculation of G -FC requires G -ADI, the applicability of this metric is restricted by two factors: (1) as already mentioned no efficient algorithm for the calculation of G -ADI can be expected to exist and (2) the extension of G -ADI to sequential circuits is problematic as the adequate handling of unreachable states is an open question.

In [J3] we proposed the following metric: It reflects whether the test set detects the resistive bridging fault f for at least one short resistance. The metric will be referred to as the *optimistic fault coverage* O -FC defined for a bridging fault f as:

$$O\text{-FC}(f) = \begin{cases} 100\% & , \text{ if } C\text{-ADI}(f) \neq \emptyset, \\ 0\% & , \text{ otherwise.} \end{cases} \quad (5.4.5)$$

As we can see, O -FC neglects the detailed resistance information provided by an ADI and only reports if the fault is detectable at all, i.e. if $C\text{-ADI}(f) \neq \emptyset$. In this sense our metric is very similar to non-probabilistic metrics common for “traditional” fault models. On the other hand, once C -ADI is known, O -FC can be calculated without any further (computational-complex) input.

All fault coverage metrics introduced so far are defined for a single bridging fault. For a set of N faults $\{f_1, f_2, \dots, f_N\}$ we calculate the average of the individual faults’ coverages. This can be obtained for each of the metrics by the equation (with ‘*’ being replaced by ‘ P ’, ‘ E ’, ‘ G ’ or ‘ O ’, as adequate):

$$*\text{-FC} = \frac{1}{N} \cdot \sum_{i=1}^N *\text{-FC}(f_i). \quad (5.4.6)$$

Note that if appropriate we do not distinguish between a metric for one single fault and the averaged version of the same metric according to Equation (5.4.6).

If $G\text{-ADI}(f_i)$ is known for every $1 \leq i \leq N$ we may exploit the fact that all shorts with $R_{\text{sh}} \notin G\text{-ADI}(f)$ are considered as being redundant. This yields a refined instance of G -FC: the *global fault efficacy* G -FE. Note that the essence of this metric is related to the concept of “fault efficiency” for single-stuck-at faults, i.e. the ratio of the number of detected faults and the number of detectable faults. The fault efficacy excludes all redundant bridging faults from consideration and focuses only on the coverage of detectable faults. The share of redundant bridging faults R , i.e. the number of faults which have an empty G -ADI, is defined as:

$$R = |\{f_i \in \{f_1, f_2, \dots, f_N\} \mid G\text{-ADI}(f_i) = \emptyset\}|.$$

Based on the number of redundant faults we can compute the global fault efficacy by modifying Equation (5.4.6) for G -FC as follows:

$$G\text{-FE} = \frac{1}{N - R} \cdot \sum_{i=1}^N G\text{-FC}(f_i). \quad (5.4.7)$$

When comparing fault coverage metrics P -FC, E -FC, and G -FC we can see that they share the same numerator. Furthermore, keeping in mind that ρ is a positive function and that $G\text{-ADI}(f) \subseteq [0, R_{\max}(f)] \subset [0, \infty]$, we observe that:

$$\int_{G\text{-ADI}(f)} \rho(r)dr \leq \int_0^{R_{\max}(f)} \rho(r)dr \leq \int_0^{\infty} \rho(r)dr.$$

Recall that $C\text{-ADI}(f) \subseteq G\text{-ADI}(f)$ and that according to Equation (5.4.5) $O\text{-FC}(f) = 100\%$, iff $C\text{-ADI}(f) \neq \emptyset$. Thus, we obtain the following relation between the fault coverage metrics introduced above:¹¹

$$P\text{-FC}(f) \leq E\text{-FC}(f) \leq G\text{-FC}(f) \leq O\text{-FC}(f). \quad (5.4.8)$$

The equation has been successfully validated by experiments we published in [J3]. Results show that (averaged over all circuits considered in the experiments) difference between P -FC and E -FC is larger than among E -FC, G -FC and O -FC. This underlines that P -FC is indeed a pessimistic metric.

The results we obtained in [J3] prove to be valuable for the calculation of G -FC, which is the preferred metric for resistive bridging faults. Since its computation is subject to limitations as discussed above, approximating G -FC may be an interesting alternative. As Equation (5.4.8) suggests, G -FC is bounded by E -FC and O -FC. Both metrics do not share the limitations of G -FC – in particular they can be calculated efficiently. Consequently, G -FC may be approximated by averaging these two fault coverages. In case the accuracy of this approximation is below the desired level, calculation of G -FC should be considered. Results from [J3] indicate that indeed E -FC and O -FC allow good approximation of G -FC. This is also supported by findings obtained with our test pattern generator for resistive bridging faults (see Chapter 8.2).

Impact of Process Variations

The electrical framework of the resistive bridging fault model is assuming fixed process parameters. In practice, these parameters are subject to statistical process variations (also referred to as parametric variations) which, amongst others, may affect transistor length L , gate oxide thickness, and substrate doping (see e.g. [6, 15, 127]). This may result in a deviation of the voltage characteristics and input logic thresholds with respect to the assumptions of the electrical model. As a consequence, the critical resistances, the analogue detectability intervals, and the fault coverages may vary among the circuits. In general, we expect this variation to be monotonic. This assumption entails, that if in a manufactured circuit a critical resistance exceeds the value predicted by the model, all other critical resistances computed by the model will also be likely to underestimate the values observed in that particular circuit (and vice versa). This directly translates to C -ADI and G -ADI which are bounded by critical resistances. Hence, in this scenario it is to be expected that due to process variations in the manufactured circuit C -ADI and G -ADI will either both

¹¹In general it is not valid to replace G -FC by G -FE in this equation.

Table 5.4: Resistance intervals reconverging at XOR gate.

Node	Resistance Ranges [Ω]		
	[0, 100]	[100, 200]	[200, ∞]
v	0	0	1
w	1	0	0
z	1	0	1

be larger or smaller compared to the model's predictions. Since the fault coverage metrics are defined as fractions of integrals over C -ADI and G -ADI, the monotonicity means that, in many instances the impact of process variations on the actual resistive bridging fault coverage will be limited. The validity of this scenario is still to be verified by simulation experiments and on manufactured silicon.

5.5 Fault Effect Propagation

For a given bridging fault and an assignment to the driving gates we are already able to determine for every driven gate the range of short resistances for which the gate sees a faulty logical value. We also know how to calculate the fault coverage for a given ADI. Yet, we are still missing the link between both steps, i.e. an answer to the following question: How do we determine for which range of short resistances are the fault effects not only visible at the fault site but also at the circuit's observable points? From the discussion in Chapter 5.1 we have learned that the assignments to the driving gates – which are induced by combinations of logical values at the circuit's primary inputs and/or by its internal state – impact the size of the resistance intervals obtained. Moreover, as an extensive study by Renovell et al. [149] has revealed, propagation of resistance intervals influences the resistance range covered by the resulting ADI. They found that propagation obeys complex mechanisms and that there is in general no one-to-one mapping from the interval obtained at the fault site to the ADI. In particular, the study demonstrated that intervals of resistances, reconverging at the inputs of a gate can map to non-trivial intervals at the gate's output. Consider as an example an XOR gate with inputs v and w and output z . Assume that input v observes a faulty logical value 0 for short resistances from the range $0 \Omega \leq R_{\text{sh}} < 200 \Omega$, and fault-free logical value 1 otherwise. Recall that this corresponds to $[0, 200] 0/1$. Assume furthermore, that input w sees $[0, 100] 1/0$, i.e. faulty logical value 1 for $0 \Omega \leq R_{\text{sh}} < 100 \Omega$ and the fault-free logical value 0 otherwise. Table 5.4 clarifies this, listing the nodes in column 1 and the relevant resistance ranges in columns 2 to 4. We observe that only for shorts having a resistance $100 \Omega < R_{\text{sh}} < 200 \Omega$ do both input nodes v and w assume the same logical value 0, while for all other short resistances $v \neq w$ holds. Obeying the logic function of the XOR gate we can conclude that z will see faulty logical value 0 for $100 < R_{\text{sh}} < 200 \Omega$ only and fault-free logical value 1 otherwise. Hence, we obtain $[100, 200] 0/1$ at z which is equal to the non-contiguous resistance range $\{[0, 100] \cup [200, \infty]\} 1/0$.

This example demonstrates that the logical values represented by the intervals, the covered resistance ranges, and a gate's logic function have to be taken into account during fault effect propagation. Several approaches to determine ADIs have been proposed. Motivated by [149] the *interval-based techniques* consider the full continuum of short resistances and use set operations to propagate intervals of resistances from the fault site to the circuit's observable points. The *sectioning technique* maps resistance intervals to a collection of conditional multiple-stuck-at faults to which the propagation rules of conventional Boolean fault simulation apply. ADIs can be reconstructed without a loss of accuracy from the results of the CMS@ simulation. A radically different approach has been proposed by Favalli et al. in [41]. For a given test vector and a bridging fault they assign Boolean variables to each input of a driven gate connected to any of the shorted nodes. Then they derive the logic function computed at each of the circuit's observable points (reachable from the shorted nodes) with respect to the assigned variables. The analysis of these functions yields combinations of logical values which when observed by the driven gates allow us to detect the fault. In the last step these logical values are matched with the true values interpreted according to the voltage characteristics induced by the test vector. Thereby critical resistances can be deduced which in turn form the detected ADI.

The experimental results published by Favalli et al. in [41] indicate that their technique is able to handle the ISCAS 85 [19] benchmark circuits in feasible time. Nevertheless we will restrict our discussion to the interval-based and the sectioning technique. Both approaches are more common and may easily be adapted to the standard fault simulation flow (see our publication [J3] and Chapter 7). In the following Chapter 5.5.1, we will reproduce the fundamental properties of the interval-based technique which immediately implements the interval concept of the resistive bridging fault model. Subsequently we will focus on the sectioning technique in Chapter 5.5.2. This technique proves to be more flexible than the interval-based one and allows for our high performance simulator SUPERB presented in Chapter 7. There, we will also present a detailed comparison of SUPERB with tools implementing either the interval-based or the sectioning technique. Subsequently, we will highlight our findings on the particularities implied by feedback bridging faults on fault effect propagation in Chapter 5.5.3. Furthermore, we investigate the probability of double errors induced by resistive bridging faults in Chapter 5.5.4. Finally, in Chapter 5.5.5 we prove that irrespective of technological parameters, detection of certain classes of resistive bridging faults can be guaranteed.

5.5.1 Interval-Based Technique

Interval-based techniques exploit the fact that intervals of short resistances are directly related to sets (refer again to Equations (5.1.1) and (5.1.2)). Thus, once we know the intervals and the logical values present at the inputs of a gate, we can compute the interval at its output using set operations. These operations have to respect both the logic function implemented by the gate and the logical values represented by the intervals at its inputs. Consequently, set operations tailored to each combination of logical values and gate function have to be used – for a detailed list of set operations refer e.g. to [98].

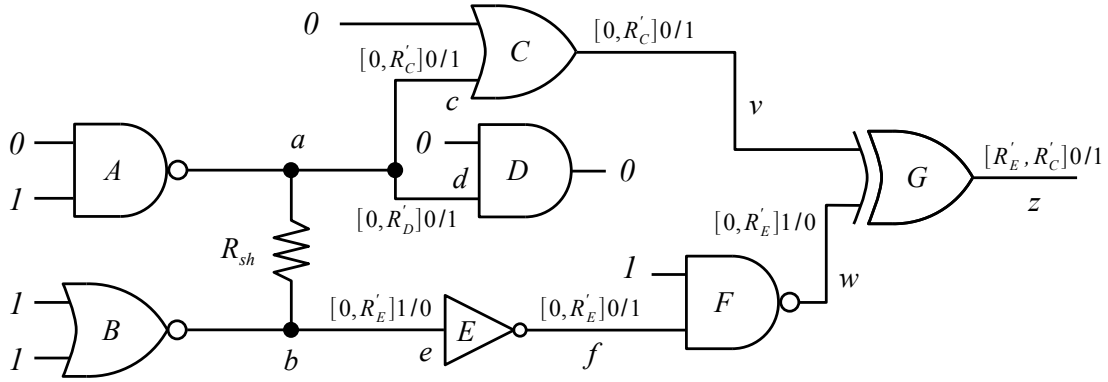


Figure 5.6: Example of interval-based simulation.

An example illustrating the propagation of resistance intervals in a small circuit is depicted in Figure 5.6. It is an extended instance of the circuit from Figure 5.1. From the analysis of the latter circuit we already know that for an input assignment $(0, 1, 1, 1)$, gates C , D and E interpret faulty logical values for intervals $[0, R'_C] 0/1$, $[0, R'_D] 0/1$, and $[0, R'_E] 1/0$, respectively. Using the abovementioned set operations these intervals can be propagated through the circuit. Observe how both the gates' logic functions as well as the logical values assigned to the non-defect affected inputs of some of the gates are respected during this process. At the circuit's outputs we obtain the interval $[R'_E, R'_C] 0/1$. In the fault-free case z would see the logical value 1, hence, the resulting ADI is $[R'_E, R'_C]$. Note that the situation at gate G exactly corresponds to the one from the example in Table 5.4 for $R'_C = 200 \Omega$ and $R'_E = 100 \Omega$.

In [J3] we managed to simplify interval propagation considerably by using a technique originally proposed by [72, p. 87]. Instead of allowing intervals to describe any logical value, e.g. $[R_1, R_2] 0/1$ and $[R_1, R_2] 1/0$, we have implicitly assumed that every interval specifies the resistance range in which the logical value 1 is present. This means that $[R_1, R_2] 0/1$ is described by $[0, R_1] \cup [R_2, \infty]$ and $[R_1, R_2] 1/0$ is represented by $[R_1, R_2]$. As a result, we do not have to consider logical values explicitly during propagation. Hence, the number of set operations required for each logic gate is reduced to one.

Our publication [J3] was the first to describe resistive bridging fault simulation for sequential circuits as well. When simulating multiple time-frames, memory elements contained in this class of circuits complicate interval propagation. Fault effects which have been stored in flip-flops may be introduced into the circuit in later time-frames. This may increase the number of reconverging intervals. When intervals are propagated to the inputs of the driving gates, this gives rise to a phenomenon we termed the *multiple strength problem*. Due to the potential overlapping of intervals present at the inputs of the driving gates the logical assignment is no longer homogeneous for the whole continuum of resistances. Consequently, the fault's activation is dependent on the short's resistance. In particular, the driving strength of both driving gates may vary with R_{sh} . This may lead to different characteristics depending on R_{sh} and likewise to different critical resistances. The multiple strength problem is not restricted to sequential circuits but may also be observed in

(combinational) circuits exposing feedback bridging faults (see Chapter 5.5.3).

In summary, the interval-based technique is indeed effective in computing ADIs. Several tools implementing this technique have been discussed in literature: see [98, 165, 166] and our publication [J3].

5.5.2 Sectioning Technique

A strategy which dramatically simplifies fault simulation and test generation for resistive bridging faults has been proposed by Shinogi et al. in [174]. We present an introduction of their technique adapted from our publications [W14] and [P15, P18]. The technique from [174] is based on two observations:

1. ADI boundaries are constituted by critical resistances only,
2. all critical resistances can be determined locally at the fault site.

This means that once we have computed all the critical resistances for a given bridging fault (which can be done very fast) we know the boundaries of all potential ADIs for this fault. Within each ADI the logical value is well-defined by definition. Consequently, the detection statuses of all shorts having a resistance from the same ADI is uniform. Either all of these shorts are detected or none are.

According to Shinogi et al., a *section* is an interval $[R_l, R_u]$ such that R_l and R_u are critical resistances, $R_l < R_u$, and there exists no other critical resistance R_c , with $R_l < R_c < R_u$. Let $R_{\text{crit}} = (R_1, R_2, \dots, R_m)$ be the list of all critical resistances, sorted in ascending order, which can be calculated for resistive bridging fault f . Then the following $m + 1$ sections exist: $[0, R_1]$, $[R_1, R_2]$, \dots , $[R_{m-1}, R_m]$, and $[R_m, \infty]$. Note that in the last section no faulty logical values are observed by any driven gate input. This means that shorts having a resistance from the range $[R_m, \infty]$ are undetectable by static logic testing, thus, we can omit this section. Occasionally we will refer to R_1 as the *minimum critical resistance* $R_{\text{min}}(f)$ of f . Inversely, R_m is referred to as the *maximum critical resistance* $R_{\text{max}}(f)$ of resistive bridging fault f .

To determine the list of all critical resistances R_{crit} for a given bridging fault f we developed Algorithm 5.4. It assumes a generalized bridging fault situation as depicted in Figure 3.2. The fault f shorts outputs a_{out} and b_{out} of driving gates A and B , respectively. Their inputs are denoted as $a_{\text{in}}^1, \dots, a_{\text{in}}^n$ and $b_{\text{in}}^1, \dots, b_{\text{in}}^m$. Driven gates are C^1, \dots, C^p connected to a_{out} , and D^1, \dots, D^q connected to b_{out} . Inputs of driven gates, which are either connected to a_{out} or b_{out} , are c_{in}^i ($1 \leq i \leq p$) for gate C and d_{in}^j ($1 \leq j \leq q$) for gate D . Initially the algorithm creates a set V (line 1) which contains all logical assignments to the inputs of A and B for which the gates' outputs a_{out} and b_{out} , respectively, assume contrary logical values (i.e. activate the bridge). Furthermore, the set of all relevant driven gate inputs G is initialized in line 2. The algorithm successively inserts critical resistances into a set R which is instantiated in line 3.

Algorithm 5.4: Generation of sorted list of critical resistances R_{crit} for bridging fault f .

Input: Bridging fault f .

Output: Sorted list of critical resistances R_{crit} .

```

/* Determine set of patterns activating  $f$  (i.e. input assignments to gates
    $A$  and  $B$  which imply  $a_{\text{out}} = 0, b_{\text{out}} = 1$  or  $a_{\text{out}} = 1, b_{\text{out}} = 0$ ). */
1  $V := \{v \in \mathbb{B}^{(n+m)} \mid v \text{ applied to } (a_{\text{in}}^1, \dots, a_{\text{in}}^n, b_{\text{in}}^1, \dots, b_{\text{in}}^m) \text{ activates } f\}$ ;
/* Determine set of inputs of driven gates  $C$  and  $D$  */
2  $G := \{c_{\text{in}}^1, \dots, c_{\text{in}}^p, d_{\text{in}}^1, \dots, d_{\text{in}}^q\}$ ;
/* Create empty set of critical resistances */
3  $R := \emptyset$ ;
4 foreach ( $v \in V$ ) do
5   foreach ( $g \in G$ ) do
6     if ( $g$  driven by pull-down for  $v$ ) then /* fault-free value is 0 */
7       Calculate  $R_{\text{crit},n}$  for  $V_{\text{lt}} = V_{\text{lt}}^g$  and  $W_n/L_n$  as induced by  $v$ ;
8        $R := R \cup \{R_{\text{crit},n}\}$ ; /* Insert  $R_{\text{crit},n}$  into  $R$  */
9     else /* fault-free value is 1 */
10      Calculate  $R_{\text{crit},p}$  for  $V_{\text{lt}} = V_{\text{lt}}^g$  and  $W_p/L_p$  as induced by  $v$ ;
11       $R := R \cup \{R_{\text{crit},p}\}$ ; /* Insert  $R_{\text{crit},p}$  into  $R$  */
12    end
13  end
14 end
15  $R_{\text{crit}} := \text{sort}(R)$ ; /* Sort set of critical resistances in ascending order. */
16 return  $R_{\text{crit}}$ ; /* Return sorted list of all critical resistances for  $f$ . */

```

Now critical resistances for all activating assignments v and all driven gate inputs g are calculated. In case g is driven by an n -transistor network, i.e. connected to the node driven to logical value 0 by v in the fault-free case, $R_{\text{crit},n}$ is calculated in line 7 (e.g. according to Equation (5.2.6)). Note that V_{lt}^g denotes the logic threshold of the driven gate input g . The resulting critical resistance is subsequently added to R (see line 8). Alternatively g is driven by a p -transistor network and the corresponding $R_{\text{crit},p}$ is computed (using e.g. Equation (5.2.7)) and added to R (lines 10 and 11).¹² Finally the set of critical resistances R is sorted in ascending order (line 15) and the resulting list R_{crit} is returned. Note that R , and consequently R_{crit} as well, do not contain any duplicate critical resistances. The total number of critical resistances for a bridging fault depends on the fault site and is bounded by two factors:

1. The maximum number of input assignments to the two driving gates. This in turn is limited by the largest number of inputs a gate from the gate library under consideration can have (multiplied by two).
2. The maximum number of gate inputs driven by the bridged nodes. This number is circuit specific, but is in general unrelated to the circuit's size.

In summary, the computational complexity of the main loop of our Algorithm 5.4 is $O(|G| \cdot |V|)$, where $|V| \leq 2^{(n+m)}$. Note that depending on the logic function implemented by the driving gates, not all possible assignments to their inputs actually activate the bridge. Hence, typically $|V|$ will be smaller than $2^{(n+m)}$. The number of critical resistances is bounded by $|G| \cdot |V|$. Likewise the number of sections is limited by the number of critical resistances.

Since within each section the logical values observed by the driven gates are well-defined, we can use conventional Boolean simulation methods to propagate potential fault effects to the circuits' observable points. Consequently, simple Boolean simulation yields the detection status of a section. Once we know the detection status of every section, we can construct the respective ADI as the union of the detected sections. Assume for instance, that for some fault f a faulty logical value can be observed for sections $[R_1, R_2]$, $[R_2, R_3]$, and $[R_4, R_5]$ while section $[R_3, R_4]$ remains undetected. The resulting ADI is thus $\{[R_1, R_3], [R_4, R_5]\}$. The segmentation of the continuum of short resistances into sections motivates the name of this approach: *sectioning technique*.

Consider again the circuit depicted in Figure 5.6 (which embeds the circuit from Figure 5.1). From the discussion in Chapter 5.1 we know that the patterns $(0, 0, 1, 1)$ and $(0, 1, 1, 1)$ activate the bridging fault. From the point of view of the critical resistance calculation covered in Chapter 5.2, $(0, 1, 1, 1)$ is equivalent to $(1, 0, 1, 1)$ due to the symmetric transistor structure of the NAND gate's pull-up network. For the sake of simplicity, we will assume that these patterns are the only valid assignments to gates A and B . From the analysis of Figure 5.2(b) – or by using Algorithm 5.4 – we know that the sorted list of critical resistances for this bridging fault is $R_{\text{crit}} = (R'_D, R_C, R'_E, R'_C, R_E)$. This corresponds to the following sections: $[0, R'_D]$, $[R'_D, R_C]$, $[R_C, R'_E]$, $[R'_E, R'_C]$, and $[R'_C, R_E]$. As already

¹²In general we will assume the values calculated using $R_{\text{crit},n}$ and $R_{\text{crit},p}$ to be natural numbers. For increased precision, rational numbers could be used as well.

Table 5.5: Section-based propagation in circuit from Figure 5.6.

Circuit node	Fault-free value	Logical value assumed in section				
		$[0, R'_D]$	$[R'_D, R_C]$	$[R_C, R'_E]$	$[R'_E, R'_C]$	$[R'_C, R_E]$
c	1	0	0	0	0	1
d	1	0	1	1	1	1
e	0	1	1	1	0	0
f	1	0	0	0	1	1
v	1	0	0	0	0	1
w	0	1	1	1	0	0
z	1	1	1	1	0	1

mentioned section $[R_E, \infty]$ does not have to be considered explicitly. For pattern $(0, 1, 1, 1)$, Table 5.5 lists the logical assignments assumed by each node (column 1), their fault-free logical value (column 2), and their faulty logical value for each section (columns 3 to 7). Comparing the logical values interpreted for instance by input c of gate C in each section with its fault-free logical value 1, we observe that the faulty logical value 0 is present in $[0, R'_D]$, $[R'_D, R_C]$, $[R_C, R'_E]$, and $[R'_E, R'_C]$. In section $[R'_C, R_E]$ (and in $[R_E, \infty]$), however, the gate derives the fault-free logical value 1. This exactly corresponds to the interval $[0, R'_C] 0/1$ associated with line c in Figure 5.6.

Logical values interpreted by gate inputs c , d , and e within one section can be propagated to output z of the circuit using the Boolean function attributed to each gate. Once we have obtained the logical value of z for all sections, we can compare this value with the expected fault-free logical value 1. It turns out that in our example the faulty logical value 0 is present in section $[R'_E, R'_C]$ only, while in all other sections the fault-free logical value 1 is attained. Hence, $[R'_E, R'_C]$ is the resulting ADI – which agrees with the result derived by the interval-based technique. This underlines that the sectioning technique involves no loss of accuracy and allows us to produce results equivalent to those of the interval-based approach.

The logical values interpreted by the driven gates for one section can be read as a multiple-stuck-at fault. Each driven gates' input sees a well-defined logical value which either matches the fault-free or the faulty logical value. In combination this corresponds to a collection of stuck-at faults. Furthermore, for one section, logical values can be propagated from the driven gates' inputs to observable points of the circuit using multiple-stuck-at fault simulation techniques. As we know, however, the values interpreted by the driven gates are not only dependent on the section but also on the assignment to the driving gates' inputs. As a consequence, each multiple-stuck-at fault is only valid if the respective assignment is present – this equals an activation condition. Hence, for one section we can represent the Boolean situation at the fault site by a conditional multiple-stuck-at fault (one for each activating assignment). To fully determine the ADI for a bridging fault which can be split into m sections, the detection statuses of at least m conditional multiple-stuck-at faults have to be evaluated. CMS@ faults for each section of our example

Table 5.6: Conditional multiple-stuck-at faults corresponding to resistive bridging faults.

Condition	Section	Multiple-stuck-at fault
0111	$[0, R'_D]$	c s-a-0, d s-a-0, e s-a-1
0111	$[R'_D, R_C]$	c s-a-0, e s-a-1
0111	$[R_C, R'_E]$	c s-a-0, e s-a-1
0111	$[R'_E, R'_C]$	c s-a-0
0111	$[R'_C, R_E]$	–
0111	$[R_E, \infty]$	–
0011	$[0, R'_D]$	c s-a-0, e s-a-1
0011	$[R'_D, R_C]$	c s-a-0, e s-a-1
0011	$[R_C, R'_E]$	e s-a-1
0011	$[R'_E, R'_C]$	e s-a-1
0011	$[R'_C, R_E]$	e s-a-1
0011	$[R_E, \infty]$	–

circuit can be found in Table 5.6 (activation conditions in column 1, sections in column 2 and multiple-stuck-at faults in column 3). For instance, in the situation in section $[R_C, R'_E]$, pattern (0, 1, 1, 1) corresponds to the following multiple-stuck-at fault: c s-a-0 and e s-a-1. Simulating this fault indeed yields $z = 1$ (as the stuck-at faults cancel each other at the XOR gate), which matches the data in Table 5.5.

Even though not explicitly discussed by Shinogi et al., the sectioning technique can also be applied to sequential circuits without modification. Thus, it is a simple and flexible way to derive ADIs for resistive bridging faults.

5.5.3 Feedback-Bridging Faults

Feedback bridging faults are a very problematic class of bridges. Due to the feedback loop created by the short and the logic path between the two circuit nodes state-holding behavior or oscillating logical values may be observed. This complicates detection of such defects. Moreover, feedback bridges constitute a non-negligible fraction of all bridging faults. A study performed by Ma et al. [105] reports that roughly 10% of all bridging fault locations they extracted from a commercial microprocessor are of the feedback type. For layouts of ISCAS 85 [19] benchmarks circuits, Chess et al. [28] found that 10% to 60% of all bridges cause feedback loops.

Consequently, many authors cover this non-trivial and frequently occurring type of bridging fault. One of the first references in this area is Mei [115] from 1974, who has identified the basic properties of feedback bridging faults. For the wired-logic models, the detectability of these faults has been theoretically analyzed by Karpovsky et al. [80, 81]. Their work, however, is restricted to feedback loops shorting a circuit's primary inputs and outputs.

Assuming the same class of bridges Xu et al. [209, 210] found that fault detection can be guaranteed using a sequence of two test patterns. Abramovici et al. [2] have extended the analysis to general classes of feedback bridging faults and report on a gate-level simulator for the wired-logic models. Their experiments demonstrate that most feedback bridging faults can be detected by a single test vector. Dahlgren [35] discusses a switch-level simulator for feedback bridging faults. Malaiya et al. [108] provided insight into the electrical conditions that impact the effects imposed by feedback loops in CMOS logic. They found the gates' propagation delays to be very influential and identified the short's resistance as a relevant factor. Rajsuman [143] has developed a technique to estimate the oscillation frequency of feedback bridging faults affecting inverter chains. The method has been generalized and refined by Hashizume et al. [64, 65].

Feedback bridging faults have also been mentioned in the context of both the voting model [3] and the biased voting model [112]. Several simulation (see e.g. [60, 146]) and test pattern generation tools (amongst others [29, 121]) have been proposed which are able to deal with non-resistive feedback bridging faults.

The combination of resistive bridging faults and feedback loops, however, has only been discussed in our publications [P3] and [J1]. We found that depending on the bridge resistance, a test vector may detect the defect, not detect the defect, or lead to oscillation. Therefore, non-contiguous ADIs are possible. Determining these ADIs can be expected to be very time consuming. To alleviate the problem we proposed a *simplified technique* in [J1]. For a given feedback bridging fault f we split the continuum of short resistances into three intervals: $[0, R_{\min}(f)]$, $[R_{\min}(f), R_{\max}(f)]$, and $[R_{\max}(f), \infty]$. Again $R_{\min}(f)$ and $R_{\max}(f)$ denote minimum and maximum critical resistance of f , respectively. Obviously, in the latter interval, no fault effects can be observed; consequently we can exclude this range from consideration. The remaining two intervals cover the range of potentially detectable short resistances. In the lowest interval $[0, R_{\min}(f)]$ the bridging fault behaves as if its resistance was 0Ω . It is rather simple to determine the detection status for this range of shorts exploiting the rich knowledge accumulated for non-resistive bridging faults. Shorts having a resistance from the lowest interval may either be detected, not detected, or lead to oscillation. As our analysis in [J1] has demonstrated, determining the detection status of shorts with a resistance from the interval $[R_{\min}(f), R_{\max}(f)]$ is possible but time consuming. To reduce computational complexity we advocate to pessimistically assume that these shorts cause oscillation. This means that observability of fault effects cannot be guaranteed. Rather it depends on the sensitivity of the automatic test equipment (ATE), whether oscillating logical values are recognized as such, or fault-free logical values are assumed instead.

To account for this impact, we proposed adapting the fault coverage metrics from Chapter 5.4 to feedback bridging faults in [J1]. We assumed, that the simulation of a feedback bridging fault f according to the simplified technique yielded two detection ranges. Let A be the C -ADI of f , i.e. the resistance range in which detection of the fault can be guaranteed. Let O be the potential detection range of f , that is the resistance range in which either detection can be guaranteed or oscillation is possible. Note that this implies $A \subseteq O$; if $A = O$ no oscillation occurs. By FC_A we denote the fault coverage obtained

for the guaranteed detection range A using Equation (5.4.3), i.e. the excitation based fault coverage E -FC.¹³ Applying the same equation to the potential detection range O yields FC_O . For $FC_A = FC_O$ no oscillation occurs. If, however, $FC_A < FC_O$ holds, the “true” fault coverage is dependent on the sensitivity of the ATE and thus bounded by FC_A and FC_O . If the probability that the ATE recognizes oscillation is known to be $k \in [0, 1]$ the “true” fault coverage FC can be defined as $FC = (1 - k) \cdot FC_A + k \cdot FC_O$. The larger the difference between FC_A and FC_O , the more important it is to understand the ATE behavior exactly.

Although in [J1] we proposed a simplified technique for interval-based simulation we can adapt the approach to the sectioning technique in a straightforward manner. We assume a three-valued simulator as can be found for instance in [167]. This type of simulator performs stuck-at fault simulation for logical values from $\{0, 1, X\}$. We use value X to indicate oscillation. Again, we only have to focus on the two intervals covering the range of potentially detectable short resistances: $[0, R_{\min}(f)]$ and $[R_{\min}(f), R_{\max}(f)]$. The former interval coincides with the section corresponding to the lowest resistance range. Its detection status can be determined very easily using techniques similar to those for interval-based simulation. The latter interval typically decomposes into several sections. They can be simulated just as in the non-feedback case by assigning X to the driven gates’ inputs. Finally, detection ranges O and A , as introduced above, are assembled according to the section’s detection status. Fault coverage may then be computed taking into account the ATE’s sensitivity.

5.5.4 Occurrence of Double Errors

Consider again the circuit from Figure 5.1 and assume we apply the pattern $(0, 0, 1, 1)$ to its inputs. From the analysis of the voltage characteristics depicted in Figure 5.2(a), we recall that both driven gates C and E interpreted a faulty logical value if $R_{\text{sh}} < R_C$ holds. Note that gate C is driven by shorted node a while gate E is connected to shorted node b . This means that for this particular situation, faulty logical values are simultaneously induced at both nodes affected by the bridge. In the following we will term this a *double error* induced by a resistive bridging fault. Expressed in a more general way: Let $C^1, \dots, C^p (D^1, \dots, D^q)$ be the successors of gate $A(B)$ driven by the shorted node $a(b)$. If at least one C^i ($1 \leq i \leq p$) and at least one D^j ($1 \leq j \leq q$) interprets a faulty logical value for some assignment to A and B and some R_{sh} this is a double error. Note that it is not sufficient if more than one successor of gate A interprets a faulty logical value but no successor of gate B (or vice versa).

Probability of double errors is relevant for the analysis of fault-tolerance techniques [78]. To be effective these techniques have to make assumptions about the number of fault locations which can be present simultaneously in a circuit. Typically, the common single fault assumption [1, p. 94] is adopted [58]. This assumption may not be valid for double errors induced by resistive bridging faults. Yet, as our analysis presented in the following

¹³Since G -ADI is not well-defined for feedback bridging faults G -FC cannot be used.

will demonstrate, the assumption of a single fault location is in general not invalidated by resistive bridging faults.

We modified the local analysis performed in Algorithm 5.4 to evaluate the occurrence of double errors and performed two experiments. In the first experiment we enumerated all possible combinations of driving and driven gates which can be constructed from a given gate library. We applied our modified Algorithm 5.4 to each combination. For our experiment we considered all gate types which are instantiated in any ISCAS 85 [19], ISCAS 89 [18], or ITC 99 [33] circuit. Furthermore, all gate types instantiated in any of the industrial circuits by NXP semiconductors (the Netherlands) were considered as well. In total, we evaluated 35,721,000 combinations. Equations from Chapter 5.2.2 were employed to calculate critical resistances. For the Shockley technology model we used parameters from the SPICE model card of a $0.35\ \mu\text{m}$ technology from AMS. We employed HSPICE with a BSIM3v3 $0.35\ \mu\text{m}$ model card to obtain the values A_p , A_n , B_p and B_n for the Fitted model. Parameters for the Predictive model were taken from the 65 nm BSIM4 model card made available by the Device Group at UC Berkeley in [22].

For the Shockley technology model, double errors were observed in 61,792 cases (equaling 0.17% of all combinations). The number of double errors for the Fitted Shockley model was 76,081 (this is 0.21%). For the Predictive model, this number was slightly higher at 455,400 (which corresponds to 1.28%). To evaluate the fault coverage impact of double errors, we performed the following experiment: For every combination of driving gates and every input assignment causing a double error we related the resistance interval in which double errors were observable to the maximum resistance range detectable for that pattern. This is very similar to fault coverage E -FC from Equation (5.4.3). Probability density function ρ as proposed in [165] was used during this experiment. This yielded an average coverage of 18.85% for the Shockley, 34.80% for the Fitted Shockley, and 2.55% for the Predictive model. Consequently, the range of short resistances for which faulty logical values are present at the fault site in case of a double error is rather small if compared to the full resistance range in which faulty logical values may be observed.

To verify that the combinations of driving and driven gates which lead to double errors actually occur in circuits, we performed a second experiment for ISCAS 85 and the combinational cores of the ISCAS 89 benchmark circuits. For each circuit we randomly selected 10,000 non-feedback bridging faults, where available.¹⁴ Subsequently, each bridge was evaluated with the same algorithm used for the first (generic) experiment. It turned out that amongst the 363,897 resistive bridging faults extracted from all circuits, some faults indeed induce double errors. The numbers we obtained were: 504 (0.14% of all bridges) for the Shockley, 146 (equaling 0.04%) for the Fitted Shockley, and 2518 (which corresponds to 0.69%) for the Predictive model. This again underlines that double errors are possible but do occur rarely. Furthermore, their coverage impact is very low.

¹⁴Note that the fault lists used in this experiment are equal to those employed in the experiments discussed in Part II of this work.

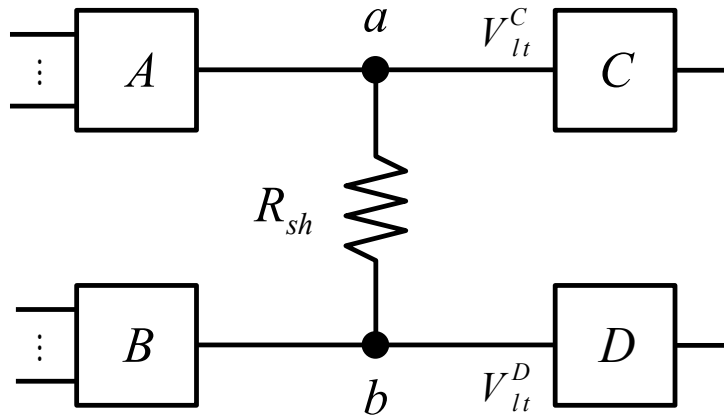


Figure 5.7: Prototype of a two successor bridge affecting nodes a and b .

5.5.5 Double Observation of Two Successor Bridging Faults

Resistive shorts have a non-trivial impact on the behavior of a circuit. Therefore we cannot expect that there are universal guidelines which guarantee the detection of all possible shorts at any fault location. Nevertheless we are able to formally derive requirements which enable us to guarantee that for a specific class of bridging faults, the fault effects of resistive shorts are observable. These are faults for which there are two driven gates and each of the single output driving gates is observed by exactly one of these gates. In the following we will refer to this class of bridging faults as *two successor bridges* (TSB). Though it might seem that this requirement is very stringent resulting in very few TSBs, this is not necessarily the case. A closer look at several sets of benchmark circuits reveals that circuit nodes with a single successor are very common and, thus, there is high potential for the existence of TSBs. The average share of circuit nodes with a single successor (with respect to all circuit nodes) is 62% for ISCAS 85 [19], 78% for ISCAS 89 [18] and 66% for ITC 99 [33] benchmark circuits.

An example for a TSB can be found in Figure 5.7. It depicts a short with resistance R_{sh} affecting outputs a and b of two arbitrary single output gates A and B . Each of the two nodes is observed by exactly one of the (disjoint) driven gates – C and D in our case. Any potential side inputs to the driven gates are omitted in the figure. The logic thresholds of gates C and D are V_{lt}^C and V_{lt}^D , respectively.

Before we provide the complete set of requirements guaranteeing the observability of fault effects and formally prove their effectiveness we first want to illustrate the general idea of the method. Consider the voltage characteristics V_a and V_b at nodes a and b depicted in Figure 5.8 for different driving strengths of gates A and B . The voltage characteristics are monotonic functions and we defined V_0 as the voltage both characteristics assume for $R_{sh} = 0\Omega$, i.e. $V_a(0) = V_b(0) = V_0$. Note that we arbitrarily presume $V_{lt}^D < V_{lt}^C$.

First assume that the fault-free logical values driven at the shorted nodes are $a = 1$ and $b = 0$ as illustrated in Figure 5.8(a). Depending on the strength of the pull-up network of gate A and the pull-down network of gate B , the voltage at a and b for $R_{sh} = 0\Omega$ can

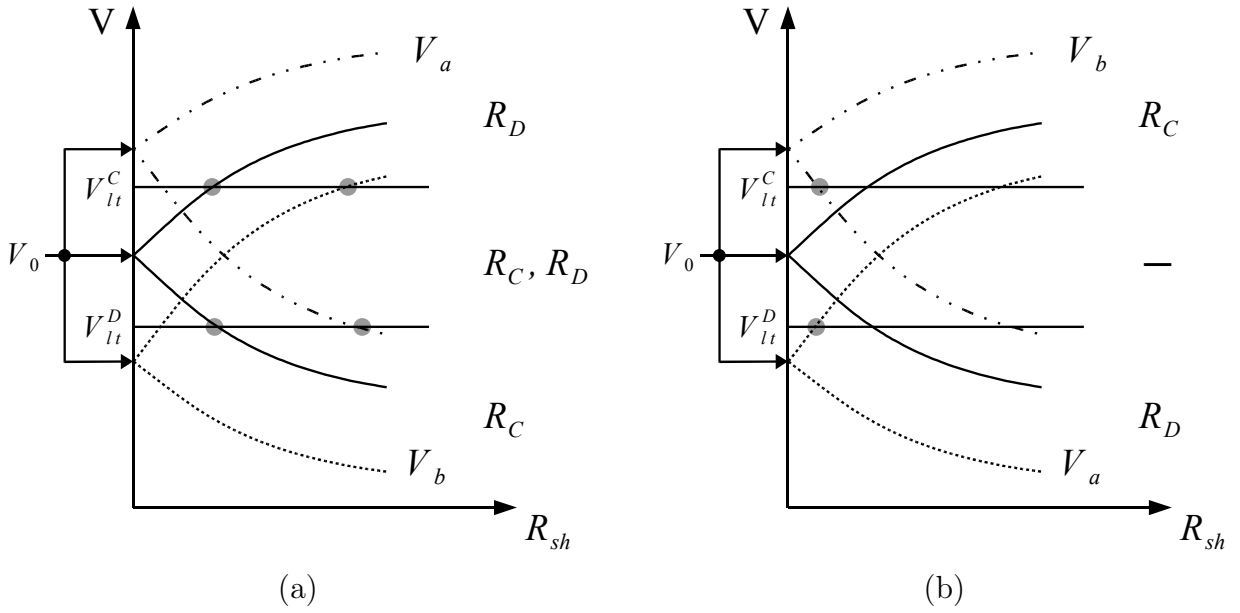


Figure 5.8: Possible voltage characteristics and critical resistances for activating assignments: (a) $a = 1$, $b = 0$, and (b) $a = 0$, $b = 1$.

attain different levels. If V_0 is larger than V_{lt}^C then there is an intersection between V_b and V_{lt}^D , yielding critical the resistance R_D for gate D . Characteristic V_a however will not cross any threshold. For $V_{lt}^D < V_0 < V_{lt}^C$ there is an intersection of both characteristics with the logic threshold of the respective driven gate. Consequently, we obtain critical resistances R_C and R_D for gates C and D , respectively. In case that $V_0 < V_{lt}^D$ there is an intersection between V_a and V_{lt}^C resulting in critical resistance R_C for gate C ; there is no intersection for characteristic V_b . Recall that the resistive bridging fault model may not unambiguously resolve critical resistances if the voltage at a bridged node equals the logic threshold of the gate observing that node. As a consequence these cases have to be excluded from consideration.

Possible voltage characteristics for $a = 0$ and $b = 1$ are depicted in Figure 5.8(b). Now the pull-up network of gate B is active, while in gate A the pull-down network is activated. For $V_0 > V_{lt}^C$ we obtain the critical resistance R_C for the input of gate C . In case $V_{lt}^D < V_0 < V_{lt}^C$ neither V_a nor V_b crosses the logic threshold of the respective driven gate. Hence, no critical resistances exist and fault detection is impossible. If $V_0 < V_{lt}^D$ we derive the critical resistance R_D for the input of gate D .

As we may observe – with one exception – at least one of the driven gates will always interpret a faulty logical value for some short resistances. This is independent of the type of the driving gates and their current input assignment. However, we have to make sure that both $a = 1$, $b = 0$ and $a = 0$, $b = 1$ are tested to rule out that fault detection is precluded by a situation like the one depicted in Figure 5.8(b) for $V_{lt}^D < V_0 < V_{lt}^C$.

Note that the essence of our observations is not invalidated if we assume $V_{lt}^D > V_{lt}^C$. Rather the occurrence of critical resistances is distributed differently for the considered cases.

Even for $V_{\text{lt}}^D = V_{\text{lt}}^C$ our observations are not invalidated. In this case it is impossible that $V_{\text{lt}}^D < V_0 < V_{\text{lt}}^C$ (or $V_{\text{lt}}^C < V_0 < V_{\text{lt}}^D$). This means in particular, that any single activating assignment to nodes a and b will cause one of the driven gates to interpret a faulty logical value.

Our observations can be summarized as follows: For a TSB, the range of short resistances for which faulty effects are observable is non-empty, provided that both activating logical assignments are applied. Neither type nor electrical properties of both driving and driven gates have to be known.

As a next step we will give a formal proof for the validity of our observation. First of all we have to establish the requirements for the existence of a critical resistance. Recall that we defined the critical resistance $R_{\text{crit}} \in \mathbb{R}_{>0}$ associated to input i of a gate connected to node n as the short resistance for which $V_n(R_{\text{crit}}) = V_{\text{lt}}^i$, where V_n is the voltage characteristic at node n and V_{lt}^i is the logic threshold of gate input i driven by n . Without loss of generality we will restrict the discussion to the input c of gate C which is connected to node a . Depending on the logical value v driven at a in the fault-free circuit, a critical resistance does exist, if:

- a) $V_{\text{lt}}^C > V_0$, if $v = 1$,
- b) $V_{\text{lt}}^C < V_0$, if $v = 0$.

Note that in case a) V_a is a monotonically increasing function, while in case b), V_a is monotonically decreasing. To prove our assertion we have to consider each case separately and make the following assumptions:

$v = 1$: Assume a critical resistance existed if $V_{\text{lt}}^C \leq V_0$. We distinguish two cases:

Case $V_{\text{lt}}^C = V_0$: The resistive bridging fault model cannot unambiguously resolve critical resistances for this case. Therefore we have to exclude it from consideration. We may, however, take a conservative position and assume that no critical resistance exists if $V_{\text{lt}}^C = V_0$.

Case $V_{\text{lt}}^C < V_0$: In this case there must be a resistance $r \in \mathbb{R}_{>0}$ such that $V_a(r) = V_{\text{lt}}^C$. Then it would hold that $V_a(r) = V_{\text{lt}}^C < V_0 = V_a(0) \Leftrightarrow V_a(r) < V_a(0)$. Yet, as $r > 0$ and V_a is a monotonically increasing function, it must be that $V_a(r) \geq V_a(0)$ and, thus, since we assumed $V_{\text{lt}}^C < V_0$, there is no critical resistance. This is a contradiction to our assumption.

$v = 0$: Assume a critical resistance existed if $V_{\text{lt}}^C \geq V_0$. We again distinguish two cases:

Case $V_{\text{lt}}^C = V_0$: We may use the same argumentation as for $v = 1$ in this case.

Case $V_{\text{lt}}^C > V_0$: There must be a resistance $r \in \mathbb{R}_{>0}$ such that $V_a(r) = V_{\text{lt}}^C$. Then it would hold that $V_a(r) = V_{\text{lt}}^C > V_0 = V_a(0) \Leftrightarrow V_a(r) > V_a(0)$. Yet, as $r > 0$ and V_a is a monotonically decreasing function it must be that $V_a(r) \leq V_a(0)$ and, thus, since we assumed $V_{\text{lt}}^C > V_0$, there is no critical resistance. Again, this is a contradiction to our assumption.

□

Inversely, we may prove in an analogous way that no critical resistance exists, if:

- a) $V_{lt}^C < V_0$, if $v = 1$,
- b) $V_{lt}^C > V_0$, if $v = 0$.

Now that we have identified the conditions for the existence of a critical resistance, we can prove the observability of TSBs. Let us define two assignments t_1 and t_2 which when applied to the inputs of our example circuit have the following properties:

- 1) vector t_1 sets $a = 0$ and $b = 1$,
- 2) vector t_2 set $a = 1$ and $b = 0$.

Again we consider a TSB like the one depicted in Figure 5.7. If for this kind of defect there is an assignment t_1 with property 1) and an assignment t_2 with property 2), then there exists at least one short resistance R_{sh} for which a faulty logical value is interpreted by at least one driven gate if t_1 and t_2 are applied. This means that if t_1 and t_2 are assigned to the inputs of the circuit, we are always able to compute a critical resistance for (at least) one of the driven gates. To prove this assertion we assume that if t_1 and t_2 are applied no driven gate interprets a faulty logical value for any value of R_{sh} and, thus, there exist no critical resistances. Depending on the logic thresholds of the driven gates we have to distinguish two cases:

Case $V_{lt}^C = V_{lt}^D$: We define $V_{lt} = V_{lt}^C = V_{lt}^D$. Since nodes a and b are at opposite values it would hold that $V_{lt} < V_0$ and at the same time $V_{lt} > V_0$ (note that we have to rule out that $V_{lt} = V_0$). Therefore, exactly one critical resistance must be obtained and we derived a contradiction to our assumption.

Case $V_{lt}^C \neq V_{lt}^D$: Depending on the fault activation it holds that:

- i) if t_1 is applied $V_{lt}^C > V_0$ and $V_{lt}^D < V_0$,
- ii) if t_2 is applied $V_{lt}^C < V_0$ and $V_{lt}^D > V_0$.

From i) we can conclude that $V_{lt}^C > V_{lt}^D$, and from ii) it follows that $V_{lt}^C < V_{lt}^D$. Since the gate's logic thresholds are fixed for all input assignments this is impossible and contradicts our assumption.

□

We have proven that the interval of short resistances for which faulty logical values are interpreted is non-empty if both t_1 and t_2 are applied. This, however, does not necessarily imply that shorts having a resistance from this interval are actually detectable. In the following we will derive a sufficient condition for the detection of TSBs. In doing so, we will make no assumptions about the type or the electrical properties of both driving and driven gates.

From the discussion we have learned that – without specific knowledge about the gates involved in the TSB – we cannot predict if fault effects may be interpreted by either gate C or gate D or both gates. In order to detect a defect we therefore have to propagate fault effects simultaneously from both nodes to an observable point. This means that we have to rephrase the requirements for t_1 and t_2 as follows:

- 1) Vector t_1 detects single-stuck-at faults a s-a-1 and b s-a-0 as well as the multiple-stuck-at fault formed by the combination of both faults.
- 2) Vector t_2 detects single-stuck-at faults a s-a-0, b s-a-1, and the combined multiple-stuck-at fault.

We have to require that both single- and multiple-stuck-at faults are detected to ensure that shorts are detected even if resistance intervals reconverging at the inputs of a gate map to non-trivial intervals at the gate's output. However, as the study introduced in Chapter 5.5.4 proved, most of the shorts cause errors originating from a single shorted node only. Therefore, we may mitigate the detection requirement by solely enforcing the detection of single-stuck-at faults. Then, however, fault detection is not guaranteed for all TSBs. A test vector set which is guaranteed to detect some defects at all two successor bridge locations has to contain for every TSB: one vector satisfying condition 1) and another vector fulfilling condition 2). Our result drastically simplifies ATPG for two successor bridges by eliminating the need to solve the equations discussed in Chapter 5.2. This also implies that the profound technological data required for these equations does not have to be provided for TSBs.

6 Summary and Discussion of Part I

In Part I of this thesis we discussed a wide range of conceptually refined models for the effects of shorts on the behavior of a logic circuit. We identified the handling of a short's intrinsic resistance as the most prominent distinction between the models under consideration. The first group, termed non-resistive bridging fault (NRBF) models, either considers this property implicitly or neglects its impact. Whereas the second group, the resistive bridging fault (RBF) model, explicitly handles the whole continuum of the intrinsic resistance. Thus, this model accounts for arbitrary resistive shorts.

We found that the non-resistive bridging fault models differ substantially in terms of the accuracy with which effects of shorts are reflected. Additionally, we explored the models' complexity. We related their complexity to the number of combinations of logic assignments to the driving gates with fault effects seen by the driven gates, that represents a single bridging fault. This number of combinations directly translates to the computational effort required when performing fault simulation or test pattern generation using the respective model.

We identified three classes of non-resistive models based on accuracy and complexity. The first class contains simple logic models like the wired-logic, the dominance-behavior, and the 4-way model. These models share a similarly low accuracy combined with low complexity. Yet, they only reflect very basic properties of shorts and completely ignore the specifics of CMOS technology. Technology-based (biased) voting models from the second class combine profound technological knowledge with an efficient voting strategy. This entails increased accuracy in conjunction with moderate complexity. The third class of models, the generalized models, favor a generic description of a short's behavior. They are not biased towards specific technological properties or defect related parameters. The unified model and the precise test generation model belong to this class. Both display extremely high complexity paired with low modeling accuracy. Apart from that, the generalized models may be advantageous whenever parameters affecting defect behavior are inaccessible or highly variable.

The extreme complexity of the generalized models has to be accepted if a short's intrinsic resistance is not considered accurately. Otherwise the bridging fault model might fail to explain fault effects caused by some resistive shorts. This is because the intrinsic resistance heavily impacts the effects caused by a short. In particular, it could be proven that a test vector which detects a short with some resistance R_1 might not be able to detect another short with a different resistance R_2 , even though both affect the same pair of circuit nodes. The resistive bridging fault model is able to accurately reflect these effects due to its parametric nature. This is made possible by a refined technological modeling

framework in combination with the dedicated consideration of the continuum of intrinsic short resistances.

The resistive bridging fault model is based on the notion of critical resistance. The critical resistance is associated with the input of a driven gate. It constitutes an upper bound for the intrinsic resistance of shorts, which are potentially detectable. Critical resistances are computed using a very flexible electrical modeling framework based on technology specific equations. We presented two novel instantiations of this framework for current technology nodes, thus underlining its versatility. Those shorts for which faulty logical values are actually seen at a circuit's observable points are identified by the analogue detectability interval (ADI). Hence, the ADI specifies the range of detectable short resistances. When obtained for a given test set, the ADI is denoted as C -ADI. We were able to prove that for two successor bridges, a non-empty C -ADI can be guaranteed under certain conditions. Accounting for these conditions drastically simplifies test pattern generation for two successor bridges. Furthermore, we reported on our recent findings which demonstrate that double errors induced by resistive bridging faults hardly occur. This result is of vital importance for the analysis of fault-tolerance techniques.

Finally, we also discussed approaches which compute the ADI given the resistance ranges in which fault effects are observable at the fault site. We addressed both feedback and non-feedback resistive bridging faults. In particular, we introduced the sectioning technique which allows the mapping of a resistive bridging fault to a collection of conditional multiple-stuck-at faults. The latter can be handled using methods known for conventional stuck-at faults. The sectioning technique is the key to the efficiency of our fault simulator SUPERB and our test pattern generator RBF-ATPG, which we will introduce in Part II of this thesis.

Similar to the idea of the sectioning technique, the fault effects attributed to a bridging fault by any other model presented in Part I can be represented as a set of CMS@ faults as well. For the generalized models this has already been elaborated on. Likewise, the aggressor/victim combinations used by the simple logic models can be represented as a collection of CMS@ faults. This holds, as they essentially extend a single-stuck-at fault by an additional constraint. Yet, for the technology based models the existence of such a mapping might not be obvious. However, both voting models establish a well-defined mapping from logical values assigned to the driving gates to a set of faulty logical values interpreted by the driven gates. This again equals a condition and a multiple-stuck-at fault, i.e. a CMS@ fault.

The most general of the models discussed in Part I is the ETFN instance of the unified model. It attributes all possible multiple-stuck-at fault combinations to each logic assignment which activates the driven gates. Every other bridging fault model can be seen as a method to identify a relevant subset of the CMS@ faults instantiated by the ETFN model.

In summary, the models presented in Part I of this thesis are substantially different in terms of accuracy and complexity. The resistive bridging fault model is superior in terms of accuracy as it explicitly considers a short's intrinsic resistance. This is complemented by only moderate complexity.

Part II

Applications of the Resistive Bridging Fault Model

7 SUPERB – A Resistive Bridging Fault Simulator

Conventional simulators for resistive bridging faults considered the continuum of short resistances as a whole. Implementations are based on interval manipulation techniques introduced in Chapter 5.5.1. They can simultaneously determine the detection status of all possible shorts at a given bridging fault location. Reports on tools of this kind have been published by Sar-Dessai et al. [165, 166] and Lee et al. [98]. Our simulator [J3] belongs to this class of tools as well. The tool from Sar-Dessai et al., implemented in the Toolkit (tcl) scripting language, can be considered a prototype. In [166] results for ISCAS 85 circuits are reported but no run-times are stated. PROBE by Lee et al. employs refined techniques for pattern-parallel simulation. Again, results for ISCAS 85 benchmarks are found in [98]. Our simulator [J3] was the first to handle sequential and combinational circuits. We have successfully applied our tool to both ISCAS 85 and ISCAS 89 benchmarks.

Unfortunately those interval-based approaches suffer from two inherent disadvantages. First of all, operations on the continuum of resistances are computationally complex. This holds, even though detection ranges are typically covered by single intervals only. Secondly, it is difficult to transfer effective speed-up techniques developed for stuck-at fault simulation – such as exploiting bit-parallelism – to interval manipulation techniques. An alternative has been put up for discussion by Shinogi et al. in [174]. They have found a way to map resistive bridging faults to conditional multiple-stuck-at faults using the sectioning technique described in Chapter 5.5.2. They also suggested extending the sectioning technique to parallel-pattern simulation. Yet in [174] they only describe a tool which implements single fault, single pattern simulation.

This chapter presents our novel Simulator Utilizing Parallel Evaluation of Resistive Bridges (SUPERB) published in [W14] and [P15, P18]. SUPERB is a resistive bridging fault simulation tool which efficiently combines sectioning technique with bit-parallel operations on machine words known from stuck-at simulation. When we first published SUPERB in [P15] there had been no implementation which actually followed the suggestion of Shinogi et al. to extend the sectioning technique to parallel-pattern simulation. At the same time, Cheung et al. [30] reported another sectioning based parallel-pattern simulator for resistive bridging faults. In contrast to SUPERB, it uses a modified version of the resistive bridging fault model which considers a range of uncertain interpretation around each logic threshold. This slightly increases the modeling complexity, yet, has not been proven to provide advantages in terms of accuracy.

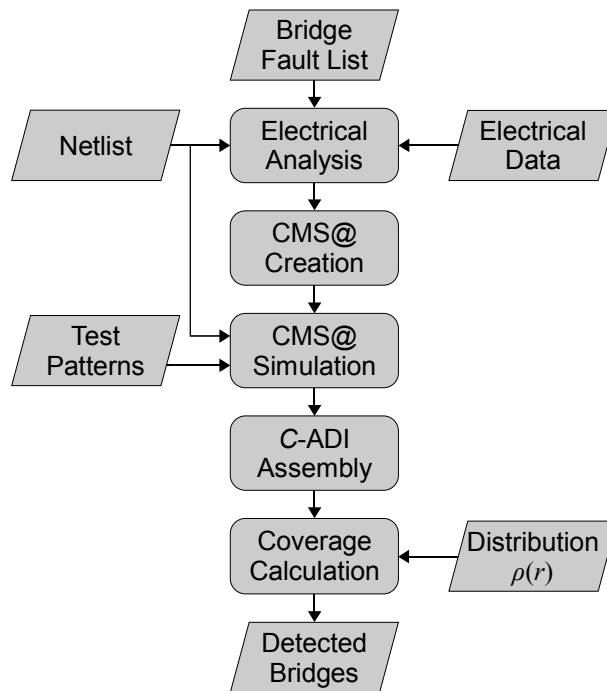


Figure 7.1: Resistive bridging fault simulation flow.

In the following, we will report on our experiments which demonstrate that SUPERB enables simulation of resistive bridging faults for multi-million gate designs in manageable time and without loss of accuracy. Furthermore, SUPERB will turn out to be more efficient than the tool from [30].

First we will introduce the outline of SUPERB in Chapter 7.1. Detailed experimental results, including a comparison of SUPERB with several competing resistive bridging fault simulators – i.e. PROBE, our tool from [J3], and the simulator from [30] – are then presented in Chapter 7.2. Finally conclusions are drawn in Chapter 7.3.

7.1 Efficient Simulation of Resistive Bridging Faults

The simulation flow of SUPERB is outlined in Figure 7.1. It can be structured into three phases. In the first phase the user specified bridging fault list is analyzed and prepared for fault simulation. First of all this requires an electrical analysis which takes the fault list, and the circuit’s gate-level netlist, the process technology, and other electrical data into account. In the analysis step, the sectioning technique as detailed in Chapter 5.5.2, is applied to each bridging fault from the list. Subsequently, the data yielded in the electrical analysis is used to create conditional multiple-stuck-at faults which are stored into an efficient data structure to allow for fast access during simulation.

In the second phase, conditional multiple-stuck-at faults corresponding to all sections of all bridging faults are simulated for a given set of test vectors. Since every section is completely

specified by its associated set of conditional multiple-stuck-at faults, no features specific to resistive bridging faults have to be taken into account in this phase. The simulation engine we have integrated into SUPERB performs a three-valued bit-parallel simulation of CMS@ faults. By considering the two logical values from \mathbb{B} and the unknown logical value, current circuit designs can be accurately simulated. Furthermore, as SUPERB is exploiting bit-parallel operations on machine words, computing resources provided by modern computers can be used efficiently. While parallel simulation techniques are well-known in the domain of single-stuck-at fault simulation, we had to extend them to conditional multiple-stuck-at faults. As a result, the efficient fault simulation engine of SUPERB enables fast and accurate simulation of resistive bridging faults.

After simulation, the detection status of each section is known. Now, in the last phase, which is again tailored to resistive bridging faults, the C -ADI of each bridging fault is assembled. If the probability density function of the short's resistance is provided, the fault coverage E -FC or G -FC is calculated (according to Equations (5.4.3) and (5.4.4), respectively). Furthermore, the list of detected resistive bridging faults may be stored upon user request. In the following we will cover phase one in more detail in Chapter 7.1.1. Phase two will be explained thoroughly in Chapter 7.1.2.

7.1.1 Fault List Preprocessing and Data Storage

For a single bridging fault the sectioning technique provides all activating assignments to the driving gates. Additionally, it yields the resistances bounding the sections and the logical values interpreted by the driven gates within each section. The critical resistances required for this procedure are identified using equations from Chapter 5.2. Alternatively they could also be obtained from electrical simulations as discussed for instance in [165]. From the section data, conditional multiple-stuck-at faults are created as explained in Chapter 5.5.2. These faults have to be quickly accessible during simulation and thus should be stored efficiently. At the same time storage requirements have to be kept low to allow processing large numbers of bridging faults in one pass.

Figure 7.2 visualizes the data structure used in SUPERB to store a single bridging fault. For illustration purposes data from Table 5.6 is used in the figure. It consists of a fault-specific *bridging fault structure* which stores general information applicable to the fault and all its sections – such as e.g. the nodes affected by the short (i.e. the fault location). Furthermore, each section is represented by its own dedicated *section structure*. Section specific information is retained here. This includes e.g. the interval covered by the respective section and its detection status.

The core of the data structure is the storage of CMS@ faults which is split into two parts. Fault activation conditions, i.e. the assignments to the driving gates which activate the bridge, are kept in a *hash table* (see e.g. [32, pp. 221]) located in the bridging fault structure. The hash table contains one entry for every assignment activating the bridge. Each assignment is mapped to a unique index. In the hash table from Figure 7.2, two

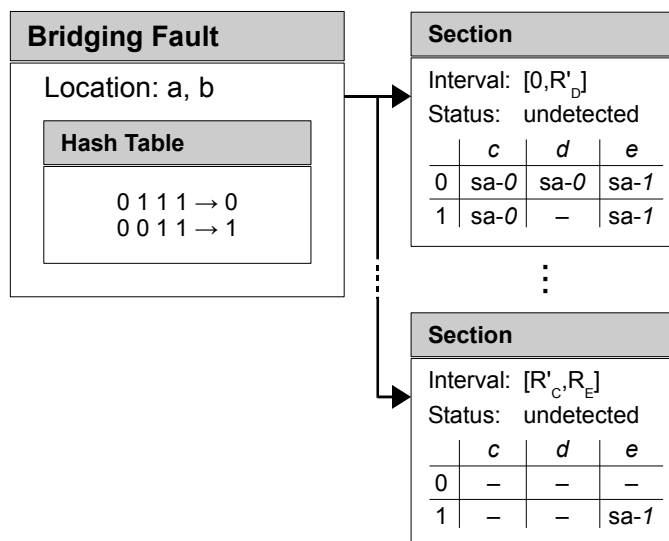


Figure 7.2: Data structure for a single bridging fault.

assignments are stored: Pattern $(0, 1, 1, 1)$ is mapped to index 0 while index 1 is allocated to pattern $(0, 0, 1, 1)$.

In contrast to the fault activation conditions, the multiple-stuck-at faults are section specific. Thus, they are kept in fault tables located separately within each section structure. Each row in a section's fault table corresponds to one multiple-stuck-at fault. The activation condition triggering this fault is indicated by the index in the first column of the table. Hence, the index links an activating assignment to the respective multiple-stuck-at fault representing the section. The individual stuck-at faults seen by the driven gates are listed in the preceding columns (in the figure “-” indicates that a gate interprets the fault-free logical value). It is important to realize that there is no one to one mapping between section and multiple-stuck-at fault. Rather, the assignment to the driven gates determines which multiple-stuck-at fault represents a section. To obtain, for instance, the multiple-stuck-at fault to be simulated for pattern $(0, 0, 1, 1)$ and section $[0, R'_D]$ (illustrated by the upper section structure in the figure) we first have to look-up the pattern's index in the hash table. Once we know this index, 1 in our example, we have to inquire the fault table belonging to the section in question. The row labeled 1 in the data structure for section $[0, R'_D]$ yields the following multiple-stuck-at fault: c s-a-0 and e s-a-1.

For efficiency reasons the keys to the hash table, i.e. the assignments to the driven gates, are represented as 32-bit unsigned integers. This is valid as the length of these keys cannot exceed $2 \cdot I_{\max}$ where I_{\max} is the maximum number of inputs of any logic gate from the gate library, and we assume I_{\max} to be less than or equal to 16. The size of the hash table, as well as the number of rows of the individual fault tables within each section structure, is bounded by $2^{2 \cdot I_{\max}}$. In practice, the tables turn out to be quite small. For each bridging fault from the fault list there is exactly one bridging fault structure. Similarly, for each section there exists one section structure, each of which belongs to exactly one bridging fault. Let n be the number of bridging faults. Then, if the number of sections for bridging

fault i , $1 \leq i \leq n$, is denoted by s_i , the total number of sections is $s = \sum_{i=1}^n s_i$. Since s_i is typically a one-digit number, the memory overhead for storing the data structure is manageable.

7.1.2 Fault Simulation Procedure

The simulation engine we have integrated into SUPERB exploits bit-parallel operations on machine words to support simulation in two different modes: parallel-pattern single-fault processing [193, 194] (PPSFP) and single-pattern parallel-fault processing [171] (SPPFP). On a machine which uses machine words of bit-width w , a bit-string B_j of length w is assigned to every node j in the circuit.¹ Note that in order to realize three-valued simulation, two bit-strings B_j^0 and B_j^1 of length w are required at every circuit node. The encoding of the three logical values used in SUPERB is equal to the one proposed by Waicukauski et al. in [194]. To simplify the discussion we will assume, however, that only the two logical values from \mathbb{B} are used. Hence, assigning a single bit-string to each circuit node is sufficient. In PPSFP mode one section is simulated for w test patterns $p_1, \dots, p_i, \dots, p_w$ in parallel. In this mode, the value of the i -th bit of bit-string B_j corresponds to the value for test pattern p_i at node j in the presence of the injected fault. In SPPFP mode w sections $s_1, \dots, s_i, \dots, s_w$, which do not necessarily belong to the same bridging fault, are simulated simultaneously for a single test pattern. In this mode, the value of the i -th bit of B_j reflects the value at circuit node j for the given pattern in the presence of the fault induced by the section s_i .

Fault injection is performed by manipulating bit masks of length w . For a given section, two masks are assigned to the inputs of all driven gates which are connected to one of the shorted nodes addressed in that section. Let j be one of these inputs. The masks assigned to j are AND mask A_j and OR mask O_j . If no stuck-at fault is present at input j , all the bits of A_j are set to the logical value 1 and all the bits of O_j are set to the logical value 0. To insert a stuck-at 0 fault at the i -th bit position of j , bit i of A_j has to be set to logical value 0. A stuck-at 1 fault is injected at bit i of input j by assigning the logical value 1 to the i -th position of O_j .

The simulation in PPSFP mode is performed using Algorithm 7.1. It takes the circuit's netlist C , a set of test patterns P , the width of a machine word w , and the set of all section structures S to be simulated as input. Note that the list of bridging faults does not have to be supplied explicitly. Initially, two bit-strings B_j and B_j' of length w are allocated for every circuit node j and the empty set of detected sections S^d is instantiated.

Starting in line 4, the outer loop of the algorithm iteratively selects w consecutive patterns from pattern set P for simulation. If the size of the pattern set $|P|$ is no integral multiple of w , less than w patterns may be selected in the last iteration of the loop (the number of patterns to be selected is determined in line 5). Subsequently *good_simulation()* is executed in line 7 which performs the parallel-pattern good-machine simulation of C . The routine assigns the patterns from P' to the bit-strings from B corresponding to the primary

¹Typically $w = 32$ or $w = 64$ for current computers.

Algorithm 7.1: PPSFP simulation of sections from S for test pattern set P .

Input: Circuit C , set of sections S , set of test patterns P , width of machine word w .

Output: Set of detected sections S^d .

```

/* Allocate bit-string of length  $w$  for each circuit node.          */
/*  $n :=$  number of nodes in  $C$ ,  $B_i \in \mathbb{B}^w$  for  $1 \leq i \leq n$     */
1  $B := (B_1, \dots, B_n)$ ; /* Good-machine copy.                      */
2  $B' := (B'_1, \dots, B'_n)$ ; /* Faulty-machine copy.              */
/* Create empty set of detected sections.                            */
3  $S^d := \emptyset$ ;
4 for ( $i := 1$ ;  $i \leq |P|$ ;  $i := i + w$ ) do
    /* Select partition of up to  $w$  patterns from  $P$                 */
    5  $k := \min(w - 1, |P| - i)$ ;
    6  $P' := \{p_j \in P \mid i \leq j \leq (i + k)\}$ ;
    /* Perform parallel-pattern good-machine simulation of circuit.  */
    7  $good\_simulation(C, B, P')$ ;
    8 foreach ( $s \in S$ ) do
    9     if ( $activated(B, s)$ ) then /* Check if  $s$  is activated by any pattern. */
        /* Init. masks for inputs of driven gates addressed in  $s$ .  */
        10  $M := create\_fault\_masks(B, s)$ ;
        /* Perform faulty-machine simulation with fault masks  $M$ .    */
        11  $fault\_simulation(C, B', M, P')$ ;
        12 if ( $obs\_difference(B, B')$ ) then /* Check if  $s$  is detected. */
            Mark  $s$  as detected;
            14  $S := S \setminus \{s\}$ ; /* Remove section  $s$  from  $S$ ... */
            15  $S^d := S^d \cup \{s\}$ ; /* ...and insert it into  $S^d$ . */
        16     end
    17     end
    18 end
19 end
20 return  $S^d$ ; /* Return set of sections detected by  $P$ .          */

```

inputs; pattern $p_i \in P'$ is assigned to the i -th bit. Then, these assignments are propagated to the observable points processing the circuit in topological order. A logic gate driving node j is simulated by applying its bit-wise logic function to the bit-strings of its inputs and then storing the resulting string in B_j .

The inner loop (starting in line 8) iteratively selects a section s from S . Note that every section contained in S gets selected exactly once in this loop. Only sections which are activated by at least one pattern from P' are simulated. This check is performed by *activated()* in line 9. To find out if a section is activated, the assignments to the driven gates induced by the patterns from P' are extracted from B . Every assignment is looked-up in the hash table located in the bridging fault structure belonging to s . Only if the inquiry of the hash table was successful and a multiple-stuck-at fault is present (for at least one pattern) in the respective row of the fault table in section s , will *activated()* be satisfied.

Prior to fault simulation, the AND and OR masks have to be initialized as explained above. This is performed by *create_fault_masks()* in line 10. Masks are initialized only for the gates driven by any of the two nodes shorted by the bridging fault represented by s . Similar to *activated()* the activating assignment at every bit position i is looked-up in the hash table. The respective multiple-stuck-at fault, which is found in the fault table in s can then be injected. Note that in contrast to regular parallel-pattern fault simulation masks O_j and A_j do not necessarily have to contain identical values. This is, because the multiple-stuck-at faults are dependent on the activation conditions induced by assignments to the driven gates.

Now, the faulty-machine simulation is conducted for the patterns from P' and section s by executing *fault_simulation()* in line 11. The algorithm is very similar to *good_simulation()* except for two differences. First of all, it modifies the faulty-machine bit-strings B' . Secondly, the driven gates' AND and OR masks from M have to be respected. If for instance a driven gate is a three-input NOR which drives node j and has inputs k , l , and m , then B'_j is obtained as

$$B'_j = \neg((B'_k \wedge A_k \vee O_k) \vee (B'_l \wedge A_l \vee O_l) \vee (B'_m \wedge A_m \vee O_m))$$

where \neg , \vee , and \wedge represent bit-wise NOT, OR, and AND operations, respectively.

After completing the fault simulation, the detection status of section s is determined by *obs_difference()* in line 12. The check simply evaluates if the bit-strings B and B' differ for any of the circuit's observable points. If so, s is marked as detected and the section is removed from set S and inserted into the set of detected sections S^d . After having simulated all sections from S for all patterns from P , Algorithm 7.1 terminates returning the detected sections S^d .

For single-pattern parallel-fault simulation, the Algorithm 7.1 has to be slightly modified. Instead of selecting w test patterns only one pattern p is simulated in the outer loop (line 4). Inversely the inner loop injects w sections $s_1, \dots, s_i, \dots, s_w$ (line 8) which are activated by test pattern p . As a consequence, *activated()*, *create_fault_masks()*, and *obs_difference()* have to be adapted such that they are able to process a list of sections instead of a single

section. This basically means repeating the steps described for the PPSFP algorithm for several sections $s_1, \dots, s_i, \dots, s_w$ but a single pattern p .

The interval-based simulator PROBE employs a technique called “pseudo-PPSFP” in [98]. Similar to SUPERB, the state of each circuit node is evaluated for several test patterns in one simulation pass. Yet, fault effect propagation does not use any bit-parallelism, rather, for each pattern, a node’s state is represented by an interval. To avoid repetitive computations, the intervals generated are stored in a list. Whenever an interval is to be created, which has already been computed, the respective entry can simply be referenced, avoiding computational complex interval operations.

7.2 Experimental Results

We applied 10,000 random test vectors to ISCAS 85 [19] circuits, the combinational cores of ISCAS 89 [18] and ITC 99 [33] benchmarks and industrial circuits by NXP semiconductors. We used the Shockley technology model and parameters from the SPICE model card of a $0.35\ \mu\text{m}$ technology from austriamicrosystems AG (AMS) to determine critical resistances (refer again to Chapter 5.2.2). Experimental results for SUPERB in 64-bit parallel-pattern simulation mode are reported in Tables 7.2, 7.3, 7.4, and 7.5, respectively.

The first column of the tables states the circuit name. It is followed by the total number of cells contained in its combinational core, and the number of primary inputs and outputs. The largest circuit considered is the industrial 2.5 million gate design p2927k. Column five of the tables lists the number of bridging faults considered in the experiments. Where available we randomly selected $10 \cdot c$ faults, c being the number of cells of the circuit as given in column two. Alternatively, we could also use layout extracted bridging faults (refer to Chapter 3.3). The number of faults was chosen to be close to the size of fault lists obtained with layout-based extraction methods.

The total number of sections can be found in column six. Depending on the benchmark suite we observed on average between 1.75 and 5.73 sections per fault. The maximum number of sections for any fault was 69. Fault coverage $E\text{-FC}$, according to Equation (5.4.3), is given in column seven. We used the distribution ρ proposed in [165] which is based on data from [158]. Subsequently, we quote the time (in seconds) needed for preprocessing the fault list (as described in Chapter 7.1.1) in column eight, the simulation time in column nine (using Algorithm 7.1) and the time required to simulate one fault for one vector in column ten (the latter is given in milliseconds). For B bridging faults, V test vectors and total simulation time T (in seconds), the normalized time per bridging fault T_{BV} calculates as:

$$T_{BV} = \frac{T \cdot 10^3}{B \cdot V}. \quad (7.2.1)$$

For comparison purposes we also performed single-stuck-at fault simulation with the same simulation engine integrated into SUPERB and the identical set of 10,000 random test

vectors. Number of single-stuck-at faults, stuck-at fault coverage and simulation time are stated in columns 11 to 13. Averages can be found in the last row of the table. We ran all experiments on a 2.8 GHz AMD Opteron Linux machine with 16 GB random access memory (RAM). SUPERB is implemented in C++.

From the tables we can observe that bridging fault coverage E -FC for 10,000 random patterns tends to be smaller for larger circuits. There are, however, also some medium-sized circuits – such as e.g. p77k – which seem to have many random-pattern resistant faults [40] and consequently display low E -FC. For many circuits we found the bridging fault coverage to be larger than the single-stuck-at fault coverage. Yet, this cannot be generalized, since for some circuits, such as p78k and p378k, the exact opposite is the case.

Time required for the preprocessing phase is very small. For the large industrial circuits, less than nine seconds on average were needed to process one million bridging faults. The largest simulation time of roughly eight hours was observed for the 2.5 million gate design p2927k. For the industrial circuits bridging fault simulation took on average 16 times longer than single-stuck-at simulation. For benchmark sets with smaller circuits this ratio was higher. Yet, we have to take into account that the average number of bridging faults exceeds the number of stuck-at faults by a factor of five. In fact the number of multiple-stuck-at faults simulated is even higher: Each bridging fault contains several sections (on average 2.6 for the industrial circuits) for each of which at least one conditional multiple-stuck-at fault has to be simulated.

Figure 7.3 shows the normalized simulation time per bridging fault as a function of the circuit size (both plotted on a logarithmic scale). It can be seen that, except for two outliers, the time per fault is almost independent of the circuit size. Figure 7.4 plots bridging fault simulation time against single-stuck-at simulation time on a logarithmic scale. The dependency is almost perfectly linear, with the exception of the smallest circuits.

Comparison with Competing Resistive Bridging Fault Simulation Tools

Table 7.6 compares SUPERB in 32-bit parallel-pattern simulation mode with the sectioning based simulator by Shinogi et al. [174], the interval-based tool PROBE [98], and our interval-based simulator [J3]. Column one of the table states the name of the ISCAS 85 benchmark circuit. Subsequently, the number of bridging faults, total simulation time in seconds and simulation time per bridging fault in milliseconds (according to Equation (7.2.1)) are quoted for the abovementioned tools. Note that the run-times for the tool by Shinogi et al. and PROBE were reported in minutes in the respective original papers. The number of test vectors was 10,000 for the tool by Shinogi et al., SUPERB, and our simulator [J3]. PROBE simulated 10,016 vectors. Total simulation time for all circuits and average T_{BV} , respectively, can be found in the last row of the table.

SUPERB is approximately two orders of magnitude faster than the tool from Shinogi et al., three orders of magnitude faster than our simulator [J3], and five orders of magnitude faster than PROBE. Shinogi et al. ran their simulator on an 850 MHz Pentium III, which may be three to five times slower than the machine we used. Moreover, the average number of sections for the circuits from Table 7.6 was 3.61 for the tool from Shinogi et al. and

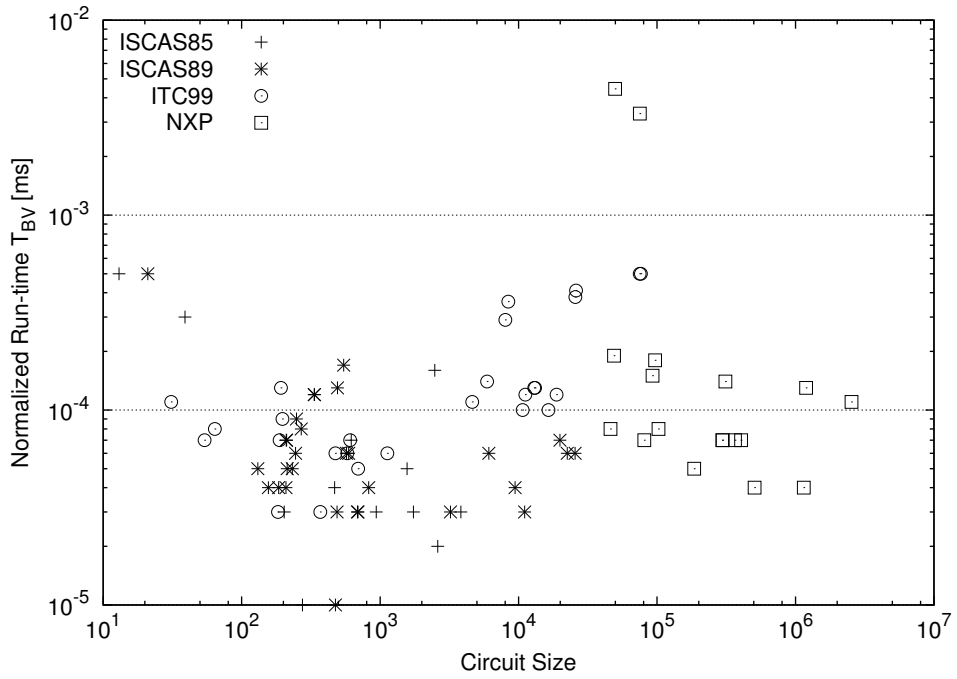


Figure 7.3: Simulation time per resistive bridging fault as function of circuit size.

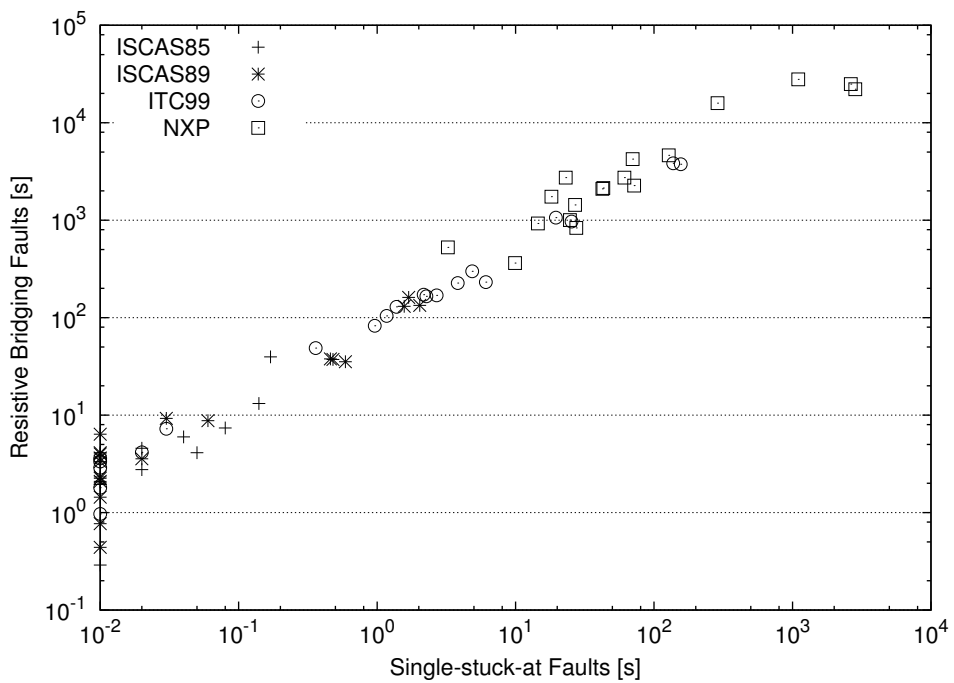


Figure 7.4: Simulation time for resistive bridging faults vs. single-stuck-at faults.

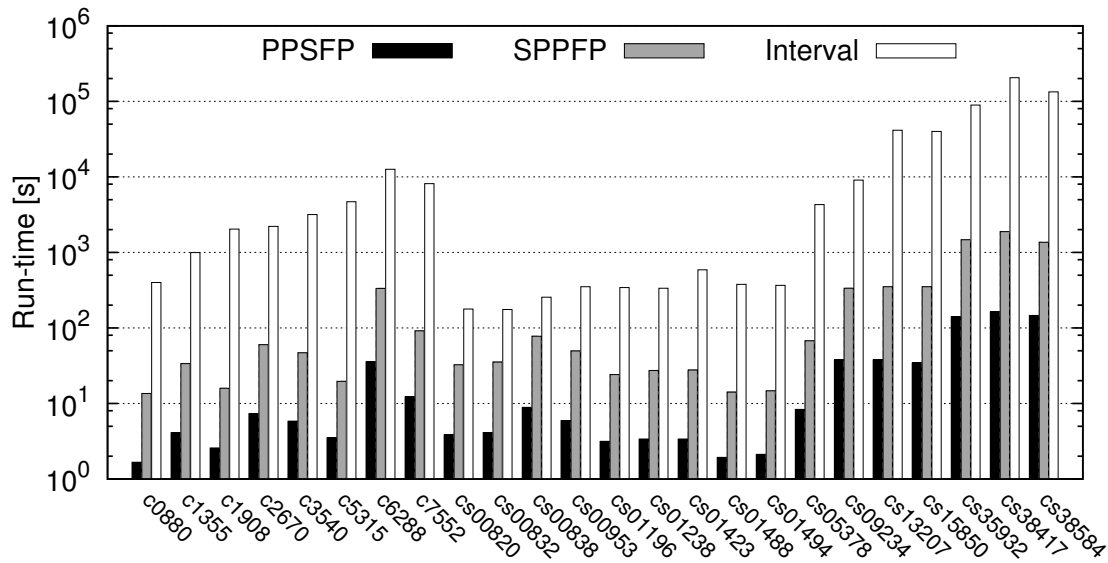


Figure 7.5: Run-time of SUPERB vs. interval-based approach from [J3].

3.11 for SUPERB, which may account for some 15% of the acceleration. The remaining difference in run-time may be predominantly attributed to the bit-parallelism employed by SUPERB and its efficient implementation. The experiments in [98] were performed on a Sun SPARC 5 with an unspecified clock frequency which may explain a slowdown of at most two orders of magnitude. The remaining difference in run-time must be attributed to the interval-based simulation technique which could not be accelerated sufficiently by the tool's “pseudo-PPSFP” approach. In contrast to the aforementioned tools, results for the simulator from [J3] were obtained on the same machine, for identical fault lists², and the same set of test vectors. Thus, the data is directly comparable. Consequently, run-time comparison with this tool underlines the superiority of the sectioning technique for the simulation of resistive bridging faults. Even though PROBE and the tool from [J3] employ the same basic simulation technique, run-times differ substantially. This might be explained to some degree by the improved representation of intervals used in the latter fault simulator.

Figure 7.5 gives further insight into the difference in performance between interval- and sectioning-based technique. For a selection of ISCAS 85 and ISCAS 89 benchmark circuits, the figure compares run-times for the interval-based simulator from [J3] and the 32-bit version of SUPERB in both parallel-pattern and parallel-fault mode. All simulations were performed on the same machine for identical fault lists and the same 10,000 random test vectors; run-times are plotted on a logarithmic scale. Taking all ISCAS 85 and ISCAS 89 benchmark circuits together (including those not contained in the figure) average run-times were 13,406.98 seconds for the interval-based simulator. For SUPERB we observed 16.44 seconds in parallel-pattern, and 163.24 seconds in parallel-fault mode. Comparing SUPERB

²Note that unless otherwise specified, the fault lists used in this chapter are equal to those employed in all other experiments discussed in Part II of this work.

Table 7.1: Comparison with the parallel-pattern resistive bridging fault simulator from Cheung et al. [30].

Circuit	Cheung et al. [30]				SUPERB (PPSFP)			
	Faults	Vectors	Time [s]	T_{BV} [ms]	Faults	Vectors	Time [s]	T_{BV} [ms]
c0432	9,132	157	7	0.00488	5,253	10,000	1.24	0.00002
c0499	16,681	149	5	0.00201	8,985	10,000	0.76	0.00001
c0880	81,899	207	21	0.00124	10,000	10,000	3.70	0.00004
c1355	90,165	399	39	0.00108	10,000	10,000	7.17	0.00007
c1908	307,416	557	11,219	0.06552	10,000	10,000	2.97	0.00003
c2670	171,603	581	38	0.00038	10,000	10,000	4.69	0.00005
c3540	206,839	540	305	0.00273	10,000	10,000	3.18	0.00003
c5315	156,699	299	68	0.00145	10,000	10,000	1.34	0.00001
c6288	191,363	61	389	0.03332	10,000	10,000	13.77	0.00014
c7552	163,137	271	75	0.00170	10,000	10,000	3.29	0.00003
\sum, \emptyset			12,166	0.01143			42.11	0.00004

in pattern-parallel mode with interval-based simulation we can observe an average speedup of 800. Parallel-fault simulation is one order of magnitude slower than parallel-pattern simulation. This is supported by results obtained for 64-bit parallel-fault simulation and the same setup used for PPSFP mode. Here, run-times of SUPERB were on average 4004.71 seconds for the ITC 99 benchmark set and 63,339.08 seconds for the industrial circuits. This again corresponds to an acceleration of one order of magnitude. The interval-based simulator [J3] did not terminate in practical time for neither ITC 99 benchmarks nor industrial circuits.

It can be observed in Figure 7.5, that for many circuits, simulation times in PPSFP-mode track the run-times of the interval-based simulator. There are, however, exceptions such as cs1238 and cs1494. Run-times reported by SUPERB in pattern-parallel mode are 3.36 and 2.11 seconds, respectively. For the interval-based tool run-times were 335.88 and 366.46 seconds, respectively. This discrepancy can be explained by the number of sections targeted by SUPERB. Our simulator [J3] considers intervals of short resistances which are typically contiguous. Hence, simulation times are hardly affected by the number of distinct critical resistances. While the average number of sections for all ISCAS 89 circuits is 2.99, SUPERB creates on average 3.41 sections for cs1238, which is 14% above the average. For cs1494 only 1.97 sections had to be targeted, which is 34% below the average.

Table 7.1 compares SUPERB in 32-bit parallel-pattern simulation mode with the sectioning based simulator by Cheung et al. [30] which also employs parallel-pattern simulation. The outline of the table is similar to the one of Table 7.6, except that we also quote the number of vectors simulated by both tools. We opted to repeat the results of SUPERB for (up to) 10,000 faults and vectors in the table to ensure a balanced comparison with the number of faults and patterns processed by the tool from [30]. The results clearly show that in terms of the run-time, the latter tool is far less efficient than SUPERB. Even if we remove

c1908 – which seems to be an outlier – from consideration, simulation of the remaining circuits still takes 947 seconds. This is 20 times more than the 42.11 seconds consumed by SUPERB for the full list of circuits. Direct comparison of these numbers seems to be justifiable as the 2.4 GHz Intel Core2 should have very similar performance than our 2.8 GHz AMD Opteron. Average T_{BV} also underlines our conclusion. To a certain degree, the difference in performance may be attributed to the large number of sections targeted by the simulator from [30] (on average 5.38 vs. 3.35 for SUPERB). Furthermore, Cheung et al. consider a range of uncertain detection. This might impose more stringent detection conditions and could thus result in less faults being dropped during simulation. However, despite this, results do underline the superiority of SUPERB.

7.3 Conclusions

This chapter has introduced SUPERB, our novel, fast simulator for resistive bridging faults. The tool combines the sectioning technique, i.e. a lossless mapping of resistive bridging faults to conditional multiple-stuck-at faults, with bit-parallel simulation and efficient data structures. This allows use to use acceleration techniques which are known from conventional stuck-at fault simulation. Experimental results underline that SUPERB is several orders of magnitude faster than conventional interval-based simulators while achieving identical accuracy. Comparison with competing sectioning based simulators confirms the superiority of SUPERB. Due to its performance, the tool enables simulation of large industrial circuits for faults lists of realistic size. Simulation of resistive bridging faults is on average one order of magnitude slower than that of single-stuck-at faults. Yet, for larger circuits the difference in simulation time tends to diminish.

Table 7.2: Experimental results for SUPERB, ISCAS 85 circuits, and 10,000 test vectors.

Circuit	Cells	PI	PO	Resistive bridging fault simulation						Stuck-at fault simulation		
				Faults	Sections	E -FC	Preproc. [s]	Sim. [s]	T_{Bv} [ms]	Faults	FC	Time [s]
c0017	13	5	2	2	8	98.59	0.01	0.00	0.00050	22	100.00	0.00
c0095	39	5	7	77	441	95.90	0.03	0.20	0.00030	110	95.45	0.00
c0432	203	36	7	2,030	7,022	98.59	0.06	0.50	0.00003	524	99.24	0.00
c0499	275	41	32	2,750	4,919	98.14	0.04	0.25	0.00001	758	98.94	0.01
c0880	469	60	26	4,690	16,977	97.78	0.07	1.87	0.00004	942	99.79	0.01
c1355	619	41	32	6,190	20,433	97.50	0.08	4.49	0.00007	1,574	99.49	0.02
c1908	938	33	25	9,380	30,338	99.35	0.09	2.67	0.00003	1,879	99.52	0.02
c2670	1,566	233	140	15,660	44,340	96.65	0.12	7.26	0.00005	2,747	88.31	0.08
c3540	1,741	50	22	17,410	49,222	98.81	0.31	5.67	0.00003	3,428	95.83	0.04
c5315	2,608	178	123	26,080	73,894	99.65	0.31	3.81	0.00002	5,350	98.90	0.05
c6288	2,480	32	32	24,800	83,190	91.65	0.22	39.33	0.00016	7,744	99.56	0.17
c7552	3,827	207	108	38,270	122,133	99.04	0.38	12.79	0.00003	7,550	94.29	0.14
∅						97.64	0.14	6.57	0.00011		97.44	0.05

Table 7.3: Experimental results for SUPERB, combinational cores of ISCAS 89 circuits, and 10,000 test vectors.

Circuit	Cells	PI	PO	Resistive bridging fault simulation					Stuck-at fault simulation			
				Faults	Sections	E-FC	Preproc. [s]	Sim. [s]	T_{BV} [ms]	Faults	FC	Time [s]
cs00027	21	7	4	2	10	100.00	0.01	0.00	0.00050	32	100.00	0.00
cs00208	131	18	9	1,310	3,911	95.70	0.01	0.71	0.00005	217	100.00	0.00
cs00298	156	17	20	1,560	4,514	97.65	0.02	0.66	0.00004	308	100.00	0.00
cs00344	210	24	26	2,100	5,734	94.01	0.02	1.35	0.00007	342	100.00	0.00
cs00349	211	24	26	2,110	5,858	94.13	0.01	1.43	0.00007	350	99.43	0.00
cs00382	209	24	27	2,090	7,698	98.60	0.02	0.75	0.00004	399	100.00	0.01
cs00386	185	13	13	1,850	3,529	96.77	0.03	0.63	0.00004	384	100.00	0.00
cs00400	213	24	27	2,130	8,119	98.30	0.04	0.94	0.00005	424	98.58	0.00
cs00420	269	34	17	2,690	7,812	92.46	0.03	2.22	0.00008	455	86.59	0.01
cs00444	232	24	27	2,320	8,952	98.07	0.03	1.20	0.00005	474	97.05	0.00
cs00510	249	25	13	2,490	10,565	95.04	0.03	2.12	0.00009	564	100.00	0.00
cs00526	245	24	27	2,450	8,310	97.46	0.05	1.39	0.00006	553	99.64	0.01
cs00641	476	54	43	4,760	8,321	99.43	0.05	0.39	0.00001	467	98.29	0.01
cs00713	489	54	42	4,890	9,664	98.41	0.03	1.31	0.00003	581	92.08	0.00
cs00820	336	23	24	3,360	13,159	94.48	0.09	3.91	0.00012	850	98.47	0.01
cs00832	334	23	24	3,340	13,279	94.70	0.08	4.03	0.00012	870	96.67	0.01
cs00838	545	66	33	5,450	16,073	76.68	0.05	9.22	0.00017	931	64.34	0.03
cs00953	492	45	52	4,920	22,179	94.36	0.07	6.30	0.00013	1,079	97.68	0.01
cs01196	593	32	32	5,930	19,647	97.13	0.06	3.27	0.00006	1,242	97.58	0.01
cs01238	572	32	32	5,720	19,532	97.00	0.04	3.53	0.00006	1,355	92.47	0.02
cs01423	827	91	79	8,270	23,767	97.07	0.07	3.56	0.00004	1,515	98.75	0.01
cs01488	692	14	25	6,920	13,589	97.94	0.06	2.03	0.00003	1,486	99.80	0.01
cs01494	686	14	25	6,860	13,480	98.12	0.07	2.25	0.00003	1,506	99.00	0.01
cs05378	3,221	214	228	32,210	92,857	99.17	0.23	8.53	0.00003	4,603	98.18	0.06
cs09234	6,094	247	250	60,940	133,205	95.56	0.34	36.92	0.00006	6,927	84.55	0.48
cs13207	9,441	700	790	94,410	191,222	98.09	0.55	37.15	0.00004	9,815	90.92	0.46
cs15850	11,067	611	684	110,670	222,797	98.26	0.61	34.67	0.00003	11,725	91.57	0.59
cs35932	19,876	1,763	2,048	198,760	540,150	96.40	1.40	132.28	0.00007	39,094	89.81	2.03
cs38417	25,585	1,664	1,742	255,850	665,758	96.00	1.61	159.89	0.00006	31,180	88.75	1.69
cs38584	22,447	1,464	1,730	224,470	588,931	95.87	1.87	129.01	0.00006	36,303	94.52	1.57
∅				96.10			0.25	19.72	0.00008	95.16		0.23

Table 7.4: Experimental results for SUPERB, combinational cores of ITC 99 circuits, and 10,000 test vectors.

Circuit	Cells	PI	PO	Resistive bridging fault simulation						Stuck-at fault simulation		
				Faults	Sections	E -FC	Preproc. [s]	Sim. [s]	T_{BV} [ms]	Faults	FC	Time [s]
b01	54	7	7	472	1,878	96.80	0.02	0.29	0.00007	122	100.00	0.00
b02	31	5	5	103	340	92.79	0.02	0.09	0.00011	62	100.00	0.00
b03	183	34	34	1,830	6,900	98.57	0.04	0.58	0.00003	394	100.00	0.00
b04	694	77	74	6,940	26,380	98.68	0.06	3.28	0.00005	1,540	99.35	0.01
b05	608	35	70	6,080	26,387	98.52	0.07	4.08	0.00007	1,554	98.78	0.02
b06	64	11	15	364	1,518	96.30	0.02	0.28	0.00008	140	100.00	0.00
b07	476	50	57	4,760	19,020	98.38	0.06	2.83	0.00006	1,129	96.90	0.01
b08	192	30	25	1,920	8,281	96.19	0.03	2.38	0.00013	417	99.76	0.00
b09	188	29	29	1,880	6,254	94.11	0.03	1.23	0.00007	414	100.00	0.00
b10	197	28	23	1,970	9,587	97.32	0.03	1.77	0.00009	486	100.00	0.01
b11	579	38	37	5,790	25,212	98.21	0.08	3.52	0.00006	1,436	99.30	0.01
b12	1,127	126	127	11,270	46,370	98.31	0.17	7.08	0.00006	2,827	93.92	0.03
b13	370	63	63	3,700	14,889	99.32	0.05	0.92	0.00003	801	100.00	0.01
b14_1	4,624	277	299	46,240	205,399	97.75	0.62	48.17	0.00011	12,475	92.63	0.36
b14	5,923	277	299	59,230	264,644	96.85	0.95	81.57	0.00014	16,167	88.87	0.96
b15_1	8,422	484	518	84,220	340,942	77.92	1.28	298.12	0.00036	22,060	60.83	4.86
b15	8,026	485	519	80,260	334,187	89.56	1.16	230.41	0.00029	21,282	73.64	6.10
b17_1	25,983	1,449	1,509	259,830	1,053,806	79.58	3.95	1059.33	0.00041	67,861	62.08	19.56
b17	25,719	1,451	1,511	257,190	1,074,642	87.30	4.03	963.14	0.00038	68,207	71.10	25.36
b18_1	74,881	3,307	3,293	748,810	3,264,135	89.24	12.03	3744.42	0.00050	202,888	75.81	155.91
b18	76,513	3,307	3,293	765,130	3,327,674	89.14	12.01	3826.46	0.00050	206,812	75.80	137.65
b20_1	11,199	522	512	111,990	513,879	97.93	1.70	127.44	0.00012	30,813	92.92	1.38
b20	12,991	522	512	129,910	581,664	97.67	2.00	163.44	0.00013	35,731	91.65	2.27
b21_1	10,696	522	512	106,960	485,942	98.49	1.55	102.62	0.00010	29,155	94.39	1.17
b21	13,168	522	512	131,680	589,635	97.51	1.88	167.47	0.00013	36,058	90.48	2.69
b22_1	16,416	735	725	164,160	741,831	98.21	2.26	169.65	0.00010	44,835	93.48	2.17
b22	18,789	735	725	187,890	841,580	97.89	2.90	223.87	0.00012	51,341	92.01	3.82
∅						94.76	1.81	416.09	0.00016		90.51	13.49

Table 7.5: Experimental results for SUPERB, combinational cores of NXP circuits, and 10,000 test vectors.

Circuit	PI			PO			Resistive bridging fault simulation				Stuck-at fault simulation			
	Cells	PI	PO	Faults	Sections	E -FC	Preproc.	Sim.	T_{BV}	Faults	FC	Time		
							[s]	[s]	[ms]			[s]		
p35k	48,927	2,912	2,229	489,270	1,064,807	82.44	3.50	922.17	0.00019	67,733	58.68	14.49		
p45k	46,075	3,739	2,550	460,750	1,176,702	97.74	3.55	360.13	0.00008	68,760	93.04	9.91		
p77k	75,033	3,487	3,400	750,330	1,762,689	78.50	6.14	24,942.20	0.00332	120,348	59.51	2,635.85		
p78k	80,875	3,148	3,484	808,750	2,258,109	97.85	7.00	519.27	0.00007	163,310	100.00	3.24		
p81k	96,722	4,029	3,952	967,220	3,020,792	87.17	12.81	1,728.02	0.00018	204,174	69.27	18.14		
p89k	92,706	4,683	4,557	927,060	2,299,118	92.00	7.96	1,426.09	0.00015	150,538	75.89	26.91		
p100k	102,443	5,902	5,829	1,024,430	2,657,269	98.28	8.88	824.15	0.00008	162,129	93.57	27.46		
p141k	185,360	11,290	10,502	1,853,600	4,455,693	98.02	14.77	987.11	0.00005	282,428	91.50	24.71		
p267k	296,404	17,332	16,621	2,964,040	6,667,733	97.10	20.49	2,075.97	0.00007	366,871	90.41	42.35		
p269k	297,497	17,333	16,621	2,974,970	6,687,225	97.12	21.14	2,115.01	0.00007	369,055	90.59	43.00		
p295k	311,901	18,508	18,521	3,119,010	7,108,620	90.80	24.28	4,214.95	0.00014	472,022	77.63	70.04		
p330k	365,492	18,010	17,468	3,654,920	8,610,620	96.12	29.20	2,706.34	0.00007	540,758	86.66	61.06		
p378k	404,367	15,732	17,420	4,043,670	11,297,429	97.96	35.81	2,702.50	0.00007	816,534	100.00	23.02		
p388k	506,034	25,005	24,065	5,060,340	12,546,388	98.87	42.40	2,223.50	0.00004	881,417	96.06	71.84		
p469k	49,771	635	403	497,710	2,268,240	98.43	8.74	22,091.80	0.00444	142,751	98.53	2,840.19		
p951k	1,147,491	92,027	104,747	11,474,910	25,291,473	99.01	93.78	4,535.13	0.00004	1,557,914	95.32	127.63		
p1522k	1,193,824	71,414	68,035	11,938,240	27,085,556	93.26	99.36	15,775.47	0.00013	1,697,662	80.91	287.23		
p2927k	2,539,052	101,844	95,143	25,390,520	57,465,542	96.57	184.06	27,668.16	0.00011	3,527,607	88.56	1,100.29		
\emptyset						94.29	34.66	6,545.44	0.00052		85.90	412.63		

Table 7.6: Comparison of SUPERB with other simulation tools for resistive bridging faults.

Circuit	Shinogi et al. [174]			PROBE [98]			Interval-based [J3]			SUPERB (PPSFP)		
	Faults	Time [s]	T_{BV} [ms]	Faults	Time [s]	T_{BV} [ms]	Faults	Time [s]	T_{BV} [ms]	Faults	Time [s]	T_{BV} [ms]
c0432	n/a	n/a	n/a	157	780.00	0.49602	5,253	413.63	0.00787	5,253	1.24	0.00002
c0499	n/a	n/a	n/a	136	900.00	0.66071	8,985	801.15	0.00892	8,985	0.76	0.00001
c0880	1,000	18.00	0.00180	949	6,960.00	0.73223	10,000	776.71	0.00777	10,000	3.70	0.00004
c1355	1,000	114.00	0.01140	639	12,960.00	2.02493	10,000	1,555.15	0.01555	10,000	7.17	0.00007
c1908	2,000	78.00	0.00390	1,662	37,680.00	2.26353	10,000	2,072.06	0.02072	10,000	2.97	0.00003
c2670	5,000	840.00	0.01680	4,294	151,020.00	3.51138	10,000	1,252.85	0.01253	10,000	4.69	0.00005
c3540	5,000	720.00	0.01440	4,431	202,860.00	4.57089	10,000	1,720.28	0.01720	10,000	3.18	0.00003
c5315	8,000	540.00	0.00675	7,121	418,500.00	5.86760	10,000	1,549.68	0.01550	10,000	1.34	0.00001
c6288	4,000	1,800.00	0.04500	3,216	603,240.00	18.72750	10,000	4,614.88	0.04615	10,000	13.77	0.00014
c7552	13,000	2,700.00	0.02077	12,106	1,194,480.00	9.85108	10,000	1,780.01	0.01780	10,000	3.29	0.00003
Σ, \emptyset		6,810.00	0.01510		2,629,380.00	4.87059		16,536.40	0.01700		42.11	0.00004

8 RBF-ATPG – An Automatic Test Pattern Generator For Resistive Bridging Faults

Automatic test pattern generation (ATPG) for bridging faults has been targeted by several authors. Yet, only some of these publications are actually taking the intrinsic short resistance into account. Even less of them are based on the resistive bridging fault model discussed in Chapter 5. One reason for this might be that in contrast to traditional ATPG for single-stuck-at faults, it is not sufficient to just generate any detecting vector. Rather, we are interested in finding one or more test vectors which cover a bridging fault for the largest detectable resistance range, as this translates into detection of as many shorts as possible. Obviously, this requires a more refined test vector selection strategy which complicates the test generation algorithm considerably.

Several test pattern generators respect the intrinsic resistance [34, 85, 107, 166, 174]. In [P10] our co-operation partners Chen et al. have also proposed such an ATPG. BART from Cusey et al. [34] uses a mapping of bridging faults to single-stuck-at faults which is similar to the 4-way model discussed in Chapter 4.1.2. The circuit's netlist is modified to stimulate the justification of assignments to the driving gates' inputs which are likely to result in improved detection conditions. This allows them to generate test patterns using a conventional single-stuck-at test pattern generator. The bridging fault simulator E-PROOFS from [60] is employed to validate the results for a fixed bridge resistance of 1 k Ω . In [107], Maeda et al. present the automatic test pattern generator RBFTG which uses the precise test generation model (refer to Chapter 4.3.2 for an introduction of the model). The concept of their tool is to guarantee the application of all activating assignments to the driving gates of a bridge, but no electrical parameters are considered. A similar approach is pursued by Chen et al. in our joint publication [P10] (see also Chapter 4.3.1 for a discussion). They also favor a generalized logic model which, however, allows them to incorporate the logic thresholds of the driven gates to enhance detectability.

Sar-Dessai et al. proposed an ATPG system in [166] which employs the resistive bridging fault model and extends the fundamental test vector selection strategies developed in [104]. Their algorithm adapts PODEM [56] such that routines for justification and propagation emphasize the detection of the highest short resistance possible. For a bridging fault which is detected by two vectors v_1 (covering the fault for $0 < R_{sh} < R_1$) and v_2 (which covers the fault for $0 < R_{sh} < R_2$), where $R_1 < R_2$, they would generate the latter test vector v_2 . There are, however, also bridging faults for which any vector detecting the

highest value of R_{sh} is not able to cover the same fault for lower resistance ranges. These faults require one or more additional vectors to assure detection of all detectable shorts. Assume for instance that vector v_3 detects a given bridging fault for $[0, R_4]$ and another vector v_4 detects the fault for $[R_3, R_5]$, where $R_3 < R_4 < R_5$. In this case, the resistance range $[0, R_5]$ can only be covered by a test set which contains both v_3 and v_4 . Yet, the ATPG system in [166] does not generate multiple vectors for a single fault. The tool from [166] is a prototype implementation in the Toolkit (tcl) scripting language. ATPG results are available only for ISCAS 85 circuits c0432, c0499, and c0880. In [174], Shinogi et al. proposed an automatic test pattern generation system based on the sectioning technique. They have recommended the use of a conventional stuck-at ATPG approach to generate test patterns detecting each section representing a bridging fault. They do not discuss any strategies to guide the selection of sections targeted explicitly for test pattern generation. As a result, similar detection conditions imposed by different sections representing the same bridging fault are not exploited. This might lead to unnecessarily large test sets. In [174], only experimental results for fault simulation of ISCAS 85 circuits are reported (see Chapter 7.2 for a discussion). No experimental data concerning test pattern generation is published in that paper.

In the following we will introduce RBF-ATPG which we published in [W6], [P7], and [J4]. RBF-ATPG is the first automatic test pattern generator for resistive bridging faults which combines the sectioning and the interval-based technique (see Chapter 5.5 for a discussion of both techniques). Our tool is able to target both combinational and full-scan circuits. It allows test generation for arbitrary non-feedback bridges between two nodes, including ones detectable at higher bridge resistance and undetectable at lower resistance, and faults requiring more than one vector for detection. Hence, it is – to the best of our knowledge – the first ATPG which enables computation of G -ADI for designs which may not be exhaustively simulated. Our experimental results for ISCAS 85 and ISCAS 89 benchmark circuits are reported which indicate a higher efficiency of RBF-ATPG in comparison with state-of-the-art single-stuck-at ATPGs.

Inspired by the combination of sectioning and interval-based technique, Khursheed et al. [85] have recently extended the approach of our tool RBF-ATPG to circuits which support *adaptive power management* (APM). APM allows a circuit to adapt its power consumption to the current operating conditions by dynamically switching between several combinations of power supply voltage and operating frequency. Detectability of resistive bridging faults is impacted by the power supply voltage (see Chapter 9.1 for a discussion). This is exploited by [85] to assure that a circuit operates correctly for all available power supply voltages.

This chapter is structured as follows: Chapter 8.1 presents an overview of RBF-ATPG and discusses the ATPG algorithm in detail. Subsequently, Chapter 8.2 gives detailed experimental results for RBF-ATPG and compares the tool with competing test generation systems. Furthermore, the implications of manufacturing technology on the test pattern generation process are explored. We also study the resistive bridging fault coverage achievable by alternative test pattern generation strategies. Conclusions are presented in Chapter 8.3.

8.1 ATPG for Resistive Bridging Faults

Our automatic test pattern generator for resistive bridging faults RBF-ATPG combines the sectioning and the interval-based technique (refer to Chapter 5.5 for a discussion of both techniques). For a given list of target bridging faults, test patterns are generated by targeting sections of a fault. Each generated pattern is simulated using our interval-based simulator [J3] to perform *fault dropping* on the list of target faults, i.e. to remove all bridges detected by that pattern from the list. Note that our sectioning based simulator SUPERB from Chapter 7 could also be used to perform fault dropping. Yet, experimental results that will be presented in Chapter 8.2 indicate that the impact of the fault simulation engine on the total run-time is negligible. Test patterns are generated using an approach based on the Boolean satisfiability problem (SAT) which is similar to e.g. [24, 54, 97, 176, 181], and to our high-performance test pattern generator TIGUAN published in [W15] and [P20]. First the test pattern generation problem for a given section is formulated in conjunctive normal form (CNF). Subsequently, a solution to the problem, i.e. a logical assignment satisfying the CNF, may be found using a SAT solver. This solution yields a test vector, or, if no solution could be determined, the proof that no such vector exists. We employ the SAT solver Chaff from [124] which could, however, be substituted by more recent solvers such as e.g. MiraXT introduced in [101]. Our RBF-ATPG tightly integrates the SAT solver and the fault dropping engine which results in an efficient tool.

For a given bridging fault f the ATPG engine always targets the section covering the highest undetected resistance range first (we will call this the “highest section”). A test pattern generated for this section is likely (although not guaranteed) to detect the largest resistance range, i.e. the highest number of sections. Additionally, the conditional multiple-stuck-at faults representing this section are probably the most difficult to detect as they typically induce less faulty logical values at the driven gates. This strategy guides the pattern generation process such that the number of section explicitly targeted by the ATPG engine is kept as low as possible. This is illustrated in Table 8.1. It is based on data which has been obtained for a bridging fault affecting the outputs of a two-input NAND gate and a two-input NOR gate, each of which has a single successor. Critical resistances in this table have been computed with equations from Chapter 5.2.2 based on the Shockley transistor model. Column 1 of the table enumerates all activating logical assignments to the driving gates’ inputs. Columns 2–7 give the sections composable from the list of all critical resistances which could be obtained for this bridging fault situation. An “X” indicates that for the respective combination of input assignment and section at least one of the two driven gates sees a faulty logical value. In the highest section [841, 1298] (section [1298, ∞] can be excluded since it is always fault-free) only assignments 8 and 9 induce faulty logical values. A test vector detecting this section has a high probability to detect sections covering lower resistance ranges as well since both assignment 8 and 9 induce faulty logical values within all sections. This is obviously not the case for every assignment which detects e.g. section [0, 324].

Due to the flexibility of SAT based test generation we are able to target all conditional multiple-stuck-at faults representing one section of f at the same time. If successful, this

Table 8.1: Section data for resistive bridging fault shorting a two-input NAND and a two-input NOR gate. Symbol “X” indicates that any driven gate interprets a faulty logical value.

Driving Gate Assignment	Sections (Values in $[\Omega]$)					
	$[0, 324]$	$[324, 420]$	$[420, 502]$	$[502, 841]$	$[841, 1298]$	$[1298, \infty]$
0	X	–	–	–	–	–
1	X	X	–	–	–	–
2	X	X	X	–	–	–
3	X	X	X	–	–	–
4	X	X	X	X	–	–
5	X	X	X	X	–	–
6	X	X	X	X	–	–
7	X	X	X	X	–	–
8	X	X	X	X	X	–
9	X	X	X	X	X	–

yields a test vector which detects one of these multiple-stuck-at faults and consequently covers the section. Subsequently this vector is simulated for all bridging faults contained in the list of target faults. For every fault R_{sh} ranges covered by the vector are marked as detected. All faults for which the complete G -ADI has been detected are dropped. It is possible that the target fault f is not removed from the fault list, although a test vector detecting the highest section has been found. In this case the highest section of f which remained undetected is targeted next. This ensures complete detection of resistive bridging faults for which more than one vector is required to cover the whole G -ADI.

If the SAT solver determines that no logical assignment satisfying the CNF exists, it is proven that no test vector detecting the targeted section of f can be found. Hence, the section is redundant and has to be removed from G -ADI(f) (initially, G -ADI(f) is set to $[0, R_{max}(f)]$, which is the maximal, potentially detectable range). Consequently, the ATPG procedure not only yields a set of test vectors covering the G -ADI of every fault in the list of target bridging faults, but also their exact G -ADI. Since G -ADI is the prerequisite when calculating the exact fault coverage G -FC according to Equation (5.4.4), RBF-ATPG also enables the use of this metric. The only alternative method for determining G -ADI known so far is based on exhaustive simulation which is virtually unfeasible even for smaller designs.

The following chapters describe RBF-ATPG in more detail. First, we will demonstrate the ATPG procedure by means of an example circuit. The example will also underline that more than one test vector may be required to cover G -ADI. Chapter 8.1.2 gives a thorough description of the algorithm.

8.1.1 Example: Deriving Test Patterns for Circuit

Figure 8.1 depicts a circuit affected by a defect shorting nodes a and b . In the fault-free case the circuit computes the equivalence (XNOR) function ($x \equiv y$) on output e and

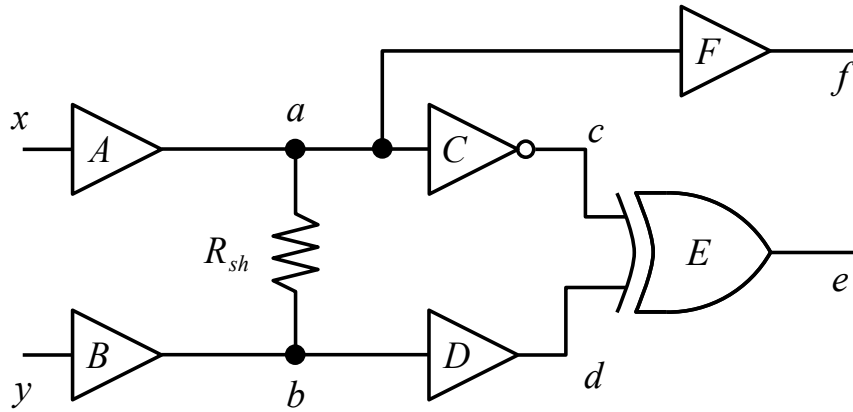


Figure 8.1: Example of circuit with a resistive short affecting nodes a and b .

propagates the value of x to output f . In the following we will first demonstrate that we need two test vectors to detect the defect for any short resistances detectable by static logic testing, i.e. to cover its complete G -ADI.¹ Afterwards we will use this circuit to illustrate the ATPG procedure.

Introduction of Circuit

Both driving gates A and B in Figure 8.1 are buffers of identical type. There are three driven gates C , D , and F . While gate C is an inverter, gates D and F are buffers of the same type (which does not necessarily agree with the type of gates A and B). In particular, this implies that the logic thresholds of gates D and F are equal, i.e. $V_{\text{lt}}^D = V_{\text{lt}}^F$. For the logic threshold of the inverter V_{lt}^C we assume $V_{\text{lt}}^C > V_{\text{lt}}^D = V_{\text{lt}}^F$. Only two assignments to inputs x and y activate the bridging fault: $(1, 0)$ and $(0, 1)$. The characteristics of the voltages induced by the two assignments on the shorted nodes a and b as a function of the short resistance R_{sh} , are depicted in Figures 8.2(a) and (b), respectively.

Consider input assignment $(1, 0)$ depicted in Figure 8.2(a). Obviously the fault-free output values are $e = 0$ and $f = 1$ for this vector. In the faulty case, using an analysis similar to the one performed in Chapter 5.1, we obtain critical resistance R_C for the input of driven gate C , and critical resistance R_D for the input of driven gate D . Since there is no intersection between characteristic V_a and the threshold V_{lt}^F of gate F (which is driven by node a) this gate does not see a faulty logical value. Consequently, we do not obtain a critical resistance in this case and no fault detection is possible on f for this vector. For $R_{\text{sh}} < R_D$, gate C interprets the logical value 0, resulting in $c = 1$. Gate D interprets the logical value 1, hence $d = 1$ and $e = 0$ (which corresponds to the fault-free logical value). For $R_D < R_{\text{sh}} < R_C$, gate C interprets the logical value 0 ($c = 1$) and D interprets the logical value 0 ($d = 0$). This results in $e = 1$; hence, for interval $[R_D, R_C]$ and assignment

¹We consider static test application only. This is in contrast to two-pattern testing required to detect delay faults (see e.g. [74]). In delay fault testing, the two vectors must be applied at-speed and their order is relevant. Neither of this is required for the detection considered here. Resistive bridging faults may also result in delay faults (see [102, 197] for details).

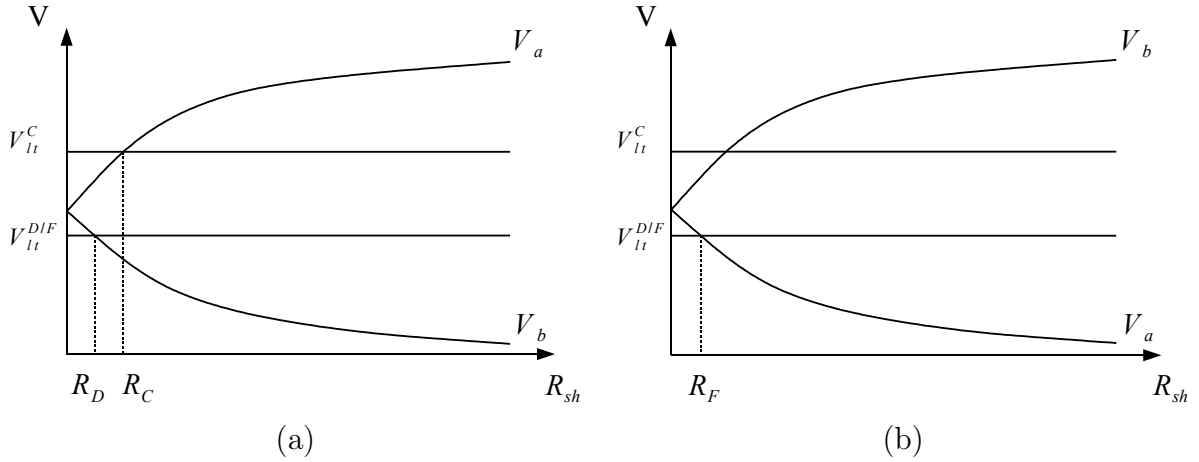


Figure 8.2: Characteristics of voltages at nodes a and b as a function of short resistance R_{sh} : (a) input assignment (1, 0) and (b) input assignment (0, 1).

(1, 0) the fault is detected on output e . For $R_{sh} > R_C$, C interprets the logical value 1, and D interprets the logical value 0, resulting in $c = d = e = 0$ and thus no detection on e .

Now consider the input assignment (0, 1) illustrated in Figure 8.2(b). For all values of R_{sh} , the voltage V_a on node a is interpreted as the logical value 0 by inverter C , and V_b on node b is interpreted as the logical value 1 by inverter D . As a consequence, output e assumes the fault-free logical value 0 independent of R_{sh} . Buffer F , however, interprets voltage V_a as the logical value 1 for $R_{sh} < R_F$ which is the faulty logical value for this vector. If $R_{sh} > R_F$, the fault-free logical value 0 is interpreted by F . Taking into account that gates D and F are of the same type, it holds that $R_D = R_F$. Therefore, we can conclude that the fault can be detected for resistance interval $[0, R_D]$ at output f for assignment (0, 1).

In summary, we derived a total of three critical resistances $R_D = R_F$, and R_C . Two sections can be composed out of these resistances: $[0, R_D]$ and $[R_D, R_C]$. Section $[0, R_D]$ is only detected by assignment (0, 1), while solely assignment (1, 0) detects section $[R_D, R_C]$. Consequently G -ADI for this fault is $[0, R_C]$ and two test vectors are necessary to completely cover this resistance range.

ATPG Procedure for Circuit

Before the SAT solver may be used to generate a test pattern for a given fault, the test pattern generation problem has to be constructed. It is based on the Boolean function bf of the fault-free and the faulty circuit. The fault-free function of our example circuit is $bf(x, y) = (e(x, y), f(x, y)) = ((x \equiv y), x)$. For section $[0, R_D]$, the function is $bf_{[0, R_D]}(x, y) = ((x \equiv y), (x \vee y))$, and for section $[R_D, R_C]$, the function is $bf_{[R_D, R_C]}(x, y) = ((x \vee \bar{y}), x)$.

According to the strategy of RBF-ATPG, the highest section is to be targeted first. Consequently the ATPG algorithm tries to generate a test vector for section $[R_D, R_C]$. This is achieved by finding a logical assignment to the circuits' inputs x and y such that

any of the components of the following function evaluate to the logical value 1:²

$$bf(x, y) \oplus bf_{[R_D, R_C]}(x, y) = ((x \equiv y) \oplus (x \vee \bar{y}), x \oplus x) = (x \wedge \bar{y}, 0).$$

This construction corresponds to a *miter circuit* (see [16]). The only assignment which sets at least one component of $bf \oplus bf_{[R_D, R_C]}$ to the logical value 1 is $x = 1$ and $y = 0$. Therefore, the test vector computed is $(1, 0)$, which matches the result we already obtained analytically.

Since test vector $(1, 0)$ is unable to cover the complete G -ADI $[0, R_C]$, the highest section which remained undetected so far has to be targeted next. In our example this is section $[0, R_D]$. A test vector detecting this interval can be determined by finding an assignment such that any of the components of the following function evaluate to logical value 1:

$$bf(x, y) \oplus bf_{[0, R_D]}(x, y) = ((x \equiv y) \oplus (x \equiv y), x \oplus (x \vee y)) = (0, \bar{x} \wedge y).$$

Again, there is only one satisfying assignment: $x = 0$ and $y = 1$. The test vector detecting section $[0, R_D]$ is thus $(0, 1)$. This also matches the results obtained in our analysis. At this point the ATPG procedure drops the fault since no undetected sections remain.

The ATPG procedure yielded two test vectors which both have to be applied in order to cover the fault's complete G -ADI. No redundant sections were identified during the procedure. Hence the G -ADI indeed equals $[0, R_C]$. The next section describes the ATPG procedure in more detail.

8.1.2 ATPG Algorithm in Detail

The main procedure of RBF-ATPG may be summarized by our Algorithm 8.1. It takes the circuit C and a set of N target faults $F = \{f_1, f_2, \dots, f_N\}$, and returns a set of test vectors V which detect G -ADI(f_i) for all faults f_i , $1 \leq i \leq N$. Furthermore, the algorithm yields G -ADI(f_i) as a byproduct.

In the preprocessing phase (starting on line 2) for every fault $f \in F$ its maximum critical resistance $R_{\max}(f)$ is computed using a modified version of Algorithm 5.4. Furthermore for each fault, two intervals G_f and L_f are initialized to $[0, R_{\max}(f)]$, which is the maximal, potentially detectable range of short resistances for fault f . Interval L_f stores all short resistances which are left to detect. Once L_f becomes empty fault, f will be dropped. When the algorithm terminates, interval G_f contains G -ADI(f).

The main outer loop, which starts on line 7, iteratively processes the set of faults F . The algorithm terminates once F becomes empty, i.e. all the faults are dropped. A fault is dropped if each of its sections is either detected by one of the test vectors generated so far or proven redundant. In every iteration of the loop a fault $f \in F$ is selected to be targeted by ATPG. Initially, the list of all critical resistances $R_{\text{crit}} = R_1, R_2, \dots, R_m$ of f is calculated in line 9 using Algorithm 5.4. From this list, all sections of fault f can be

²Note that the SAT solver operates on a CNF representation of this problem.

Algorithm 8.1: ATPG procedure for resistive bridging faults.

Input: Circuit C , set of faults $F = \{f_1, f_2, \dots, f_N\}$ **Output:** Test set $V = \{v_1, v_2, \dots\}$ that covers G -ADIs of all faults in F

```
1  $V := \emptyset$ ;  
   /* Compute the maximum critical resistance  $R_{\max}(f)$  for every fault  $f$   
   (using a modified version of Alg. 5.4) and initialize  $G_f$  and  $L_f$ .      */  
2 foreach ( $f \in F$ ) do  
3   | Compute  $R_{\max}(f)$ ;  
4   |  $G_f := [0, R_{\max}(f)]$ ;  
5   |  $L_f := [0, R_{\max}(f)]$ ;  
6 end  
7 while ( $F$  not empty) do  
8   | Select fault  $f \in F$ ;  
   /* Compute the critical resistances for  $f$  using Algorithm 5.4.      */  
9   | Compute the critical resistances for  $f$ :  $0\Omega = R_0 < R_1 < \dots < R_m < \infty$ ;  
10  repeat  
   | /* Determine section  $[R_{j-1}, R_j]$  covering the highest resistance range  
   | yet undetected.      */  
11  |  $j := \max\{k \in \mathbb{N}_{>0} \mid ([R_{k-1}, R_k] \cap L_f) \neq \emptyset\}$ ;  
12  |  $v := \text{gen\_test}(C, f, [R_{j-1}, R_j])$ ; /* Generate vector  $v$  for  $[R_{j-1}, R_j]$ .      */  
13  | if (no  $v$  found) then  
14  |   |  $G_f := G_f \setminus [R_{j-1}, R_j]$ ; /*  $[R_{j-1}, R_j]$  is redundant.      */  
15  |   |  $L_f := L_f \setminus [R_{j-1}, R_j]$ ;  
16  |   | if ( $L_f = \emptyset$ ) then  $F := F \setminus \{f\}$ ; /* Drop fault  $f$ .      */  
17  | else  
18  |   |  $V := V \cup \{v\}$ ; /* Insert  $v$  into test vector set  $V$ .      */  
19  |   | foreach ( $f' \in F$ ) do  
20  |   |   |  $D := \text{fsim}(C, f', v)$ ; /* Determine interval  $D$  detected by  $v$ .      */  
21  |   |   |  $L_{f'} := L_{f'} \setminus D$ ;  
22  |   |   | if ( $L_{f'} = \emptyset$ ) then  $F := F \setminus \{f'\}$ ; /* Drop fault  $f'$ .      */  
23  |   | end  
24  |   | end  
25  | until ( $L_f = \emptyset$ ) ;  
26 end  
27 return  $V$ ;
```

constructed as described in Chapter 5.5.2. Note that, as section data has to be maintained only for the current target fault, memory consumption of the algorithm is moderate.

In the inner loop, starting in line 10, one or more test vectors are generated until set L_f becomes empty. This loop ensures that faults requiring more than one test vector for full coverage of G -ADI are processed correctly. The section to be targeted next by the ATPG procedure is determined on line 11. As outlined before we always target the highest section $[R_{j-1}, R_j]$ undetected so far, i.e. the highest section contained in L_f .

Next, on line 12 procedure $gen_test()$ is used to generate a test vector v detecting section $[R_{j-1}, R_j]$ of f . Let circuit C have n inputs and p outputs. The procedure constructs the CNF representation of the fault-free circuit, $bf_C : \mathbb{B}^n \rightarrow \mathbb{B}^p$, and the CNF of the faulty circuit, $bf_{C,f,[R_{j-1},R_j]} : \mathbb{B}^n \rightarrow \mathbb{B}^p$. The CNF representing the Boolean function computed by the i -th output of C is denoted by bf_C^i and $bf_{C,f,[R_{j-1},R_j]}^i$, respectively. Test generation is performed by finding an assignment to Boolean variables x_1, \dots, x_n satisfying the formula:

$$\bigvee_{i=1}^p \left(bf_C^i(x_1, \dots, x_n) \oplus bf_{C,f,[R_{j-1},R_j]}^i(x_1, \dots, x_n) \right). \quad (8.1.1)$$

As mentioned above, we integrated the SAT solver Chaff into RBF-ATPG. The program library of Chaff maintains the clause database which stores the CNF and provides algorithms to solve the formula. It would be possible to replace Chaff by any other SAT solver. Furthermore, our procedure is not necessarily restricted to SAT based ATPG, and can be adapted to circuit-structure based algorithms as e.g. [49, 56, 62, 161, 168], as well.

If no solution to Equation (8.1.1) is found (lines 14 – 16), the interval $[R_{j-1}, R_j]$ is redundant and can be removed from G_f . Furthermore, the range is eliminated from L_f , and if L_f is empty, fault f is dropped. If an assignment satisfying Equation (8.1.1) could be found (lines 18 – 23) vector $v = (x_1, \dots, x_n)$ is inserted into the set of test vectors V and simulated for all faults $f' \in F$. On line 20 resistive bridging fault simulation is performed by routine $fsim()$, which takes circuit C , fault f' , and test vector v . In the current implementation of RBF-ATPG, our interval-based simulator [J3] is used. It would, however, be possible to employ our simulator SUPERB from Chapter 7, which uses the sectioning technique, as well. For every fault f' the simulator returns the ADI D covered by v . If D is non-empty, the ADI is removed from $L_{f'}$ and does not have to be targeted by ATPG in subsequent iterations. If $L_{f'}$ is empty after simulation, fault f' can be removed from the set of target faults. Consequently, not all faults from F have to be targeted explicitly by ATPG. This also ensures that the current target fault f is dropped if it has been fully covered. If at the end of this iteration interval L_f of f is non-empty, i.e. f remained undetected for some values of R_{sh} , the inner ATPG loop of the algorithm is repeated, starting on line 10.

In the (theoretical) worst case, the inner loop of the algorithm is executed m times for every fault explicitly targeted by ATPG, where m is the number of sections of that fault.³

³In the experiments discussed in Chapter 7.2 we observed that on average m is a small single digit number.

This ensures that every section of every fault in F is either covered by a test vector or proven redundant.

8.2 Experimental Results

We applied the RBF-ATPG procedure to the ISCAS 85 [19] and the combinational cores of the ISCAS 89 [18] benchmark circuits. The fault list contained 10,000 randomly selected non-feedback bridging faults, where available.⁴ Alternatively, we could also use layout extracted bridging faults (refer to Chapter 3.3). We employed the Shockley technology model and parameters from the SPICE model card of a $0.35\ \mu\text{m}$ technology from austriamicrosystems AG (AMS) to determine the critical resistances (refer again to Chapter 5.2.2). All experiments were performed on a 2 GHz Pentium IV with 2 GB main memory.

Experimental results for RBF-ATPG can be found in Table 8.4. The first column states the name of the circuit, which is followed by the number of randomly selected bridging faults in column two. Column three, labeled “UF”, gives the number of faults that were undetectable for any value of R_{sh} . Note that fault refers to a bridging fault, consequently a bridge with m sections is counted as one single fault. Subsequently the fourth column “Vec” quotes the number of test vectors obtained with RBF-ATPG. The next column contains the tools efficiency “Eff”, which is defined as the ratio of the number of faults and the number of test vectors, i.e. column two divided by column four. In column six the number of undetectable sections “US” is given. The succeeding column lists the ratio $G\text{-ADI}/[0, R_{\text{max}}]$, which assesses the impact of undetectable sections on $G\text{-ADI}$. The last two columns contain the time in seconds needed for solving all SAT instances and the total run-time of RBF-ATPG. In the table’s last row averages are given.

Each call of the SAT solver may yield one out of three possible results: (1) a solution to the problem formulated as a CNF exists. In this case a test vector was successfully generated. (2) there does not exist any solution to the problem. This means that the section for which the CNF was generated is provably redundant. (3) the SAT solver run was aborted (e.g. due to excessive memory usage). In our experiments we did not experience any aborts for the circuits quoted. As a consequence, the number of SAT solver calls equals the sum of columns “Vec” and “US”. The number of undetectable sections is high, many times it exceeds the number of test vectors generated. On average we observed 17.12% of all sections to be undetectable. The maximum number of undetectable sections is obtained for cs00386, for which 2077 out 3529 sections are undetectable, this is 58.86%. To assess the impact of the undetectable sections, we related $G\text{-ADI}$ – which contains detectable sections only – to the range $[0, R_{\text{max}}]$ of all shorts that can be activated locally at the fault site – which comprises both detectable and undetectable sections. Our following equation

⁴Note that unless otherwise specified the fault lists used in this chapter are equal to those employed in all other experiments discussed in Part II of this work.

computes the quotient of both ranges weighting each individually by the probability density function ρ to account for the occurrence probability of short resistances:

$$\frac{G\text{-ADI}}{[0, R_{\max}]} = 100\% \cdot \left(\int_{G\text{-ADI}} \rho(r) dr \right) / \left(\int_0^{R_{\max}} \rho(r) dr \right). \quad (8.2.1)$$

The numbers in Table 8.4 we determined for ρ based on the data from [178] which is in particular appropriate for modern copper interconnect technology.

Equation (8.2.1) is especially valuable to determine the accuracy to be expected when approximating $G\text{-FC}$ by $E\text{-FC}$ (Equations (5.4.4) and (5.4.3), respectively) as proposed in Chapter 5.4. Recall that $G\text{-FC}$ is based on $G\text{-ADI}$ while $E\text{-FC}$ utilizes $[0, R_{\max}]$ as a reference. If the equation evaluates to 100% this means that $G\text{-ADI}$ and $[0, R_{\max}]$ perfectly match. Hence, $E\text{-FC}$ is an equivalent substitute for $G\text{-FC}$. Values considerably smaller than 100% indicate that the approximation is less accurate. In summary, the impact of undetectable sections is less than expected given their high number. The poorest result is 85.58% (for cs00953), the best result is 99.33% (for cs13207), and the average over all circuits is 94.43%. Even the ratio of 93.97% obtained for cs00386, the circuit which featured the largest number of undetectable sections, is only slightly below the average. This may indicate that the undetectable sections are relatively small and/or cover resistance ranges with low occurrence probability. Furthermore it demonstrates that $E\text{-FC}$ provides in general a reasonable approximation of $G\text{-FC}$.

Most of the difference between the total run-time of the tool and the time needed to solve the SAT instances is consumed by CNF composition and the fault simulation runs. Yet, as on average more than 99% of the total run-time is dedicated to SAT solving, the influence of activities not related to the solving process is negligible. We believe that the tool's overall run-time may be reduced significantly by either using a SAT solver with higher performance or a powerful circuit-structure based test pattern generation algorithm.

Performance of Single-Stuck-At Test Vectors

To explore the efficacy of single-stuck-at test patterns for the detection of resistive bridging faults we performed an experiment whose results are listed in Table 8.5. Column one of the table gives the name of the circuit. The following column, labeled "RBF", states the number of test vectors obtained with RBF-ATPG for the complete set of bridging faults (quoted from column 4 of Table 8.4). Subsequently, in column three we list the number of single-stuck-at test patterns which were created by a commercial tool. The succeeding column contains the number of top-up vectors. This is the number of test patterns generated by RBF-ATPG for the list of sections which remained undetected after simulation of the single-stuck-at test patterns. In the last column we give the fault efficacy $G\text{-FE}$ (from Equation (5.4.7)) obtained when simulating only the single-stuck-at test patterns.

As can be seen, the performance of single-stuck-at test vectors is rather low – the average resistive bridging fault coverage is 96.05%. A substantial number of top-up vectors is

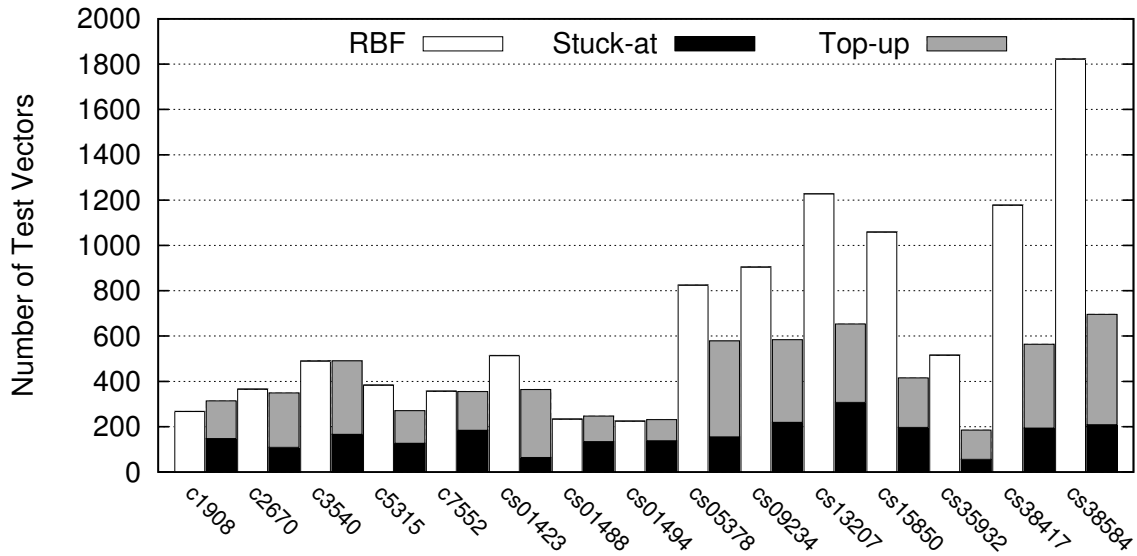


Figure 8.3: Performance of resistive bridging fault test vectors in comparison with combination of stuck-at and top-up vectors.

needed to cover the complete G -ADI and to obtain 100% coverage. On the other hand, a combination of single-stuck-at and RBF-ATPG test vectors may be beneficial in terms of the total number of test vectors. This is illustrated in Figure 8.3 where the number of test vectors generated by RBF-ATPG is plotted next to the size of the combined single-stuck-at and top-up test vector set (for a selection of circuits). In particular, for the large ISCAS 89 circuits the number of test vectors contained in the combined set is drastically smaller than what is obtained from RBF-ATPG alone – even though G -FE is 100% in both cases. This can be attributed in part to the aggressive compaction techniques which were employed when generating the single-stuck-at test sets. Currently equivalent techniques are not available in RBF-ATPG.

Comparison with Competing ATPG Systems

Table 8.2 compares RBF-ATPG with BART [34], which is based on a reduction of non-resistive bridging to stuck-at faults, and the resistive bridging fault test pattern generator from Sar-Dessai et al. [166]. Column one lists the name of the circuit. Note that in fact for s298, s344, and s641 only combinational cores were considered by all authors. In column two the efficiency “Eff” of RBF-ATPG is given (quoted from column five of Table 8.4). The remaining columns contain number of faults, vectors and the efficiency for BART and Sar-Dessai’s tool, respectively. Efficiency is again computed as number of faults divided by number of vectors. For the circuits contained in [34] the efficiency of RBF-ATPG is considerably higher. Note that higher efficiency implies that more faults are detected per vector. Comparability with Sar-Dessai’s ATPG is limited due to the small number of circuits considered. Efficiency, however, seems to be similar to that of RBF-ATPG.

Table 8.2: Efficiency of RBF-ATPG compared with BART and ATPG by Sar-Dessai et al.

Circuit	RBF-ATPG	BART [34]		Sar-Dessai's ATPG [166]			
	Eff	Faults	Vectors	Eff	Faults	Vectors	Eff
c432	7.18	1,000	542	1.85	1211	151	8.02
c499	166.39	1,000	270	3.70	1640	130	12.62
c880	13.42	1,000	622	1.61	2813	130	21.64
c1908	37.45	1,000	703	1.42	n/a	n/a	n/a
s298	35.74	1,000	200	5.00	n/a	n/a	n/a
s344	37.67	1,000	121	8.26	n/a	n/a	n/a
s641	32.89	1,000	160	6.25	n/a	n/a	n/a

We also compared RBF-ATPG with the automatic test pattern generator proposed in [P10] for the unified model (refer to Chapter 4.3.1). Table 8.3 summarizes results obtained for the ISFN_TH instance of the unified model for which data for ISCAS 89 circuits is available in [P10]. Column one and two give the circuit name and the efficiency of RBF-ATPG for the complete list of bridging faults, respectively (the latter figures are quoted from Table 8.4, column five). For the experiments, randomly selected bridging faults (different from those used by RBF-ATPG) were simulated – the number of faults is listed in column three. Test generation for the unified model employs a two-stage approach. In the first stage a single-stuck-at test set (generated by a commercial ATPG) is simulated and all faults detected by these vectors are removed from consideration. Only the remaining faults are explicitly targeted by the ATPG from [P10] in the second stage. The number of test vectors simulated in the first stage is given in column four; the number of top-up test vectors generated in stage two is found in column five. It is followed by the total size of the combined test set required to cover all bridging faults. In column seven of the table efficiency of the test pattern generator for the unified model is given. The last row of the table states average vector counts and efficiencies, respectively.

With the exception of cs09324, for all circuits the efficiency of the unified model ATPG exceeds that of RBF-ATPG. To a certain extent this may be attributed to the single-stuck-at test set employed in the first stage of the test generation process. Typically these test sets are highly compacted and, although they do not detect all bridging faults, they do allow for a basic coverage. We observed similar mechanisms when evaluating the performance of single-stuck-at test sets for resistive bridging faults listed in Table 8.5. To improve comparability we calculated the efficiency achieved with the test sets combining single-stuck-at test patterns and top-up vectors generated by RBF-ATPG. Results are found in column eight of Table 8.3. Note that the single-stuck-at test sets used for the experiments with the unified model are not the same as the ones used by RBF-ATPG. Comparison of the last two columns of the table indicates that for a comparable setup, efficiency of [P10] is similar to that of RBF-ATPG.

To further assess the efficiency of RBF-ATPG we also compare the tool with the bridging fault ATPG system RBF_{TG} from Maeda et al. [107] and with state-of-the-art single-stuck-

Table 8.3: Efficiency of RBF-ATPG compared with unified model (instance ISFN_TH).

Circuit	RBF-ATPG	Unified Model [P10]				RBF-ATPG	
	Complete Eff	Faults	Stuck-at	Top-up	Total	Eff	Top-up Eff
cs09234	11.06	5,000	209	428	637	7.85	17.12
cs13207	8.14	5,000	295	226	521	9.60	15.31
cs15850	9.43	5,000	201	234	435	11.49	24.10
cs38417	8.49	5,000	185	145	330	15.15	17.73
cs38584	5.49	5,000	191	160	351	14.25	14.37
	8.52		216.20	238.60	454.80	11.67	17.73

at ATPG systems ATOM [62] and SPIRIT [54].⁵ Maeda et al. employ locally exhaustive test generation based on the PTG model (refer to Chapter 4.3.2). Results are presented in Table 8.6. Column one of the table gives the name of the circuit. It is followed in column two by the efficiency of RBF-ATPG (quoted from Table 8.4, column five). Subsequently number of faults, number of vectors “Vec”, and the efficiency “Eff” of RBFTG are stated in columns 3–5. (Note that in [107] only results for ISCAS 89 circuits are available.) The fault lists used by Maeda et al. are based on a random selection of 1,000 bridging faults (where available) from which feedback faults were excluded. We quote the lowest number of vectors achieved by either method RDM (based on random test vectors) or ALG (based on deterministic ATPG) published in [107]. Column six of Table 8.6 gives the number of collapsed single stuck-at faults presented to both ATOM and SPIRIT. This is followed by the number of generated test vectors and the tools’ efficiency. Note that similar to RBF-ATPG both ATOM and SPIRIT perform fault dropping but do not employ any test set compaction techniques, such as reverse-order simulation (see e.g. [168]).

Even though the fault list considered by RBF-ATPG is much larger than the list used by RBFTG (up to 10,000 vs. up to 1,000 bridging faults) the number of vectors generated by RBF-ATPG is comparable and in many cases even smaller. Consequently, in comparison, the efficiency of RBFTG is very low – on average a difference of factor 10 can be observed. In comparison with both single-stuck-at ATPG systems, ATOM and SPIRIT, the number of test vectors generated by RBF-ATPG is commonly higher. However, the number of target faults considered by the bridging fault tool considerably exceeds the size of the single-stuck-at fault set (RBF-ATPG potentially targets multiple sections per bridging fault). Contrary to that, in terms of efficiency, RBF-ATPG outperforms both single-stuck-at tools for all but the largest circuits. This is somewhat counterintuitive since detection conditions imposed by the conditional multiple-stuck-at faults representing a section can be expected to be more specific than those imposed by single-stuck-at faults. Only a few test vectors might be able to fulfill these requirements and these vectors may not be useful in detecting other faults. Additionally, it is at least theoretically possible that more than one test vector is needed to detect a resistive bridging fault for the maximal resistance

⁵We are grateful to E. Gizdarski (Synopsis, USA) for providing unpublished data on the number of test vectors generated by SPIRIT.

range (see Chapter 8.1.1). On the other hand, a vector that sensitizes an output of a gate may detect several bridging faults with that output involved, but only two stuck-at faults. Furthermore, the larger number of faults targeted by RBF-ATPG increases the chance, that a vector detects multiple faults. Overall, however, it seems that the vectors that satisfy the detection conditions for the highest values of R_{sh} are also highly effective in detecting other faults.

8.2.1 Evaluation of n -Detection and 4-way Test Vectors

It is frequently argued that n -detection [106, 147] test sets are very effective in the detection of defects (see e.g. [8, 13]). These test sets contain vectors which detect each possible single-stuck-at fault at least n times (if feasible). Since resistive shorts are one particularly prominent class of defects, we should expect n -detection vectors to be extremely effective in their detection.⁶ While the n -detection paradigm targets arbitrary defects, the 4-way model (refer again to Chapter 4.1.2) is tailored to be used as a basis for ATPG for (resistive) shorts. In this chapter we want to investigate whether any of the two approaches indeed produces vectors effective in the detection of shorts. We will use the resistive bridging fault model as an indicator for the quality of the test vectors. Beyond that, the amount of vectors produced by either model will be taken into account.

We simulated n -detection and 4-way test vectors with the bridging fault simulator SUPERB introduced in Chapter 7. We focused on the large ISCAS 85 and the combinational cores of the large ISCAS 89 benchmark circuits for which n -detection test patterns from the Kyushu Institute of Technology⁷ (Japan) were made available to us. The considered values of n were 1 (i.e. conventional single detection), 5, and 10. We randomly selected 10,000 non-feedback resistive bridging faults. For these fault lists, test patterns for the 4-way model were generated using a commercial tool. The n -detection test pattern generation targets all single-stuck-at faults. Resistive bridging fault simulation was conducted for the Shockley model and the parameters from the model card of the AMS 0.35 μm technology. Fault efficacy G -FE was calculated using Equation (5.4.7) and the probability distribution ρ based on findings from [178]. All experiments were conducted on a 2.8 GHz AMD Opteron Linux machine with 4 GB RAM.

Detailed experimental results can be found in Table 8.7. Column one of the table gives the name of the circuit. Then the number “EVec” of test patterns detecting at least one short, the size “Vec” of the complete test pattern set and the global fault efficacy G -FE are quoted for all considered test pattern sets. The last row of the table contains the total number of vectors and the average fault coverage, respectively.

As can be seen, the test set sizes differ substantially. In contrast to RBF-ATPG, the tools used to generate the test vector sets under consideration employed aggressive compaction

⁶In Chapter 10.2 multiple detection of single-stuck-at faults is identified as an effective criterion which helps to maintain the coverage of resistive shorts in a BIST environment.

⁷We are grateful to Prof. X. Wen (Kyushu Institute of Technology, Japan) for providing the n -detection test pattern sets.

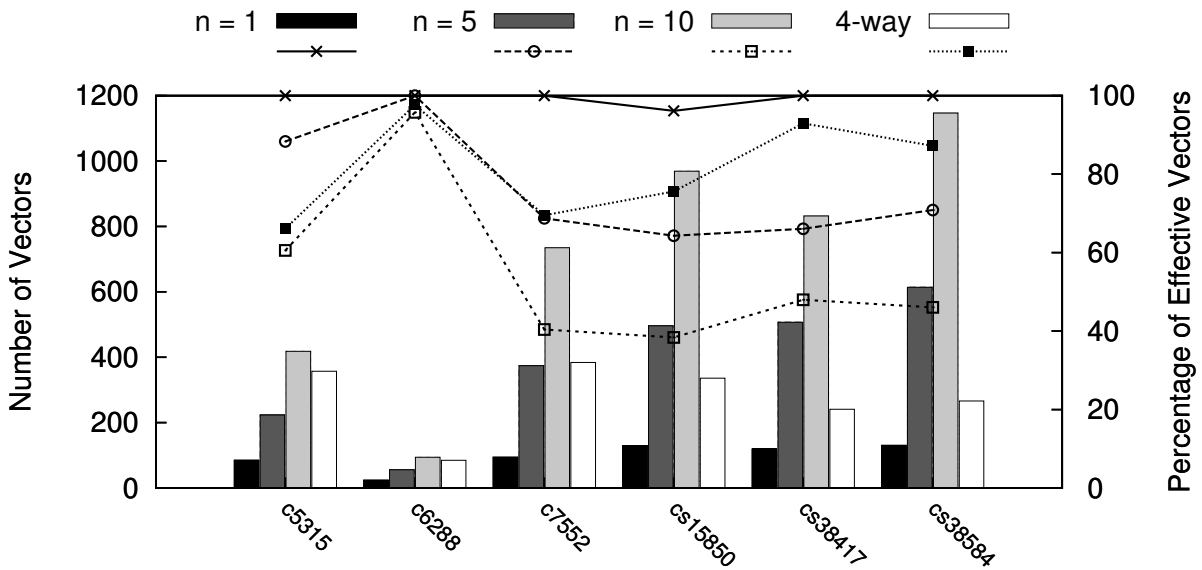


Figure 8.4: Number of test vectors (bar graph) and share of test vectors effective in detection of resistive bridging faults (line graph) for n -detection and 4-way test sets.

strategies. The smallest test sets are always obtained for $n = 1$, i.e. the regular single detection single-stuck-at scenario.⁸ For $n = 5$ test sets are considerably larger, an average increase by a factor of 3.8 may be observed. The largest test sets were generated for $n = 10$. Compared to those for $n = 1$ they contain, on average, seven times more vectors. By contrast the test sets constructed for the 4-way model are considerably smaller (factor 2.8 larger than those for $n = 1$). For the ISCAS 89 circuits, test set sizes range between the number of vectors obtained for $n = 1$ and $n = 5$, while sizes of test sets for ISCAS 85 benchmarks are comparable to those for $n = 5$. Test set sizes are depicted as a bar graph in Figure 8.4.

We also investigated the share of effective vectors, i.e. patterns that detected at least one short at any bridging fault location. With the exception of cs15850 (96.12% effective vectors), for $n = 1$ all test vectors are effective. To some degree this may be attributed to the extremely small test vectors sets. A similar effect can be observed for c6288 which features very small test vectors sets for all considered values of n and the 4-way model. For the latter model the percentage of effective vectors consistently exceeds that of the test sets for $n = 5$ and $n = 10$. The lowest share of effective vectors is found for $n = 10$, on average only 46% of the vectors detect at least one short. In Figure 8.4 this data is illustrated as a line graph.

In terms of the coverage of resistive bridging faults, test sets for $n = 1$ always yield the lowest results of, on average, 98.10%. For all other test sets, values of G -FE are very close,

⁸Note that these test sets are not the same as the ones used for the experiment detailed in Table 8.5. For this experiment an ATPG was used from which solely patterns for $n = 1$ were available. We decided to use only one test pattern source for all values of n to ensure comparability of results.

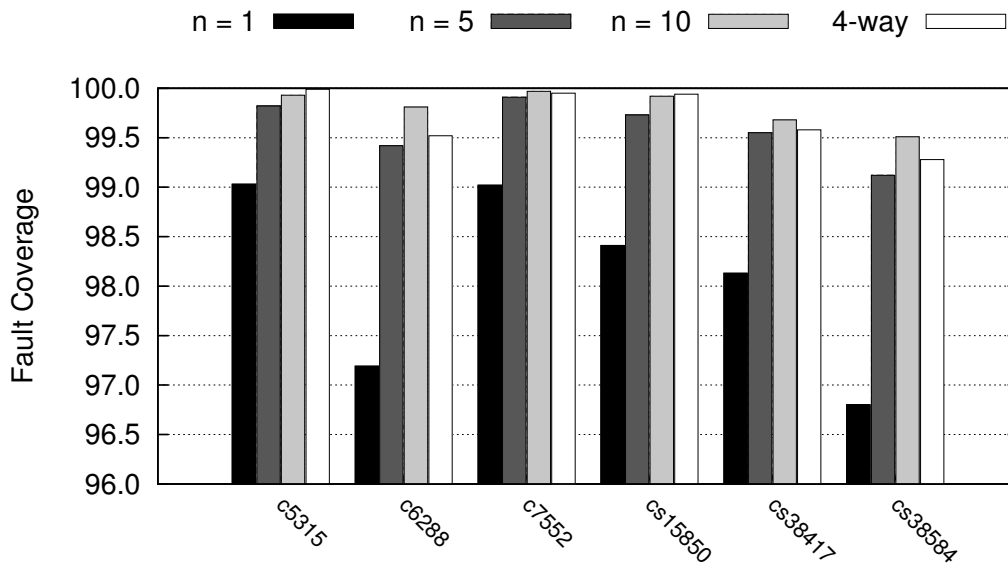


Figure 8.5: Resistive bridging fault coverage of n -detection and 4-way test sets.

ranging on average between 99.59% and 99.80% (results are summarized in Figure 8.5). The highest coverage is obtained – for the majority of circuits – for $n = 10$ followed very closely by the results for the 4-way model.

In summary, the resistive bridging fault coverages of the n -detection vectors for $n = 5$ and $n = 10$, and the 4-way model test vectors are very high, exceeding 99.0%. Furthermore all coverage figures are very close. Consequently, in terms of the coverage, no clear trend can be recognized. This picture changes when we also take the number of vectors required to achieve these results into account. Only the 4-way model features both, reasonable resistive bridging fault coverage, and small test sets. Even though the 5-detection test sets for some circuits are comparable in size, they are on average considerably larger and contain more vectors ineffective in the detection of the bridging faults under consideration.

8.2.2 Test Pattern Generation for Different Technology Models

Test pattern generator RBF-ATPG as well as SUPERB (see Chapter 7) and our simulator [J3] support three different technology models: the Shockley, the Fitted and the Predictive model (refer again to Chapter 5.2.2 for a discussion). This chapter studies the impact of these models on the test pattern generation process and the implications of *design shrinking* [73, 157], i.e. manufacturing an existing device in a next-generation technology without actual redesign, for resistive bridging fault detection.

For the experiments we used HSPICE with a BSIM3v3 $0.35\ \mu\text{m}$ model card to obtain the values A_p , A_n , B_p and B_n for the Fitted model. For the Predictive model we used the 65 nm BSIM4 model card made available by the Device Group at UC Berkeley in [22]. The

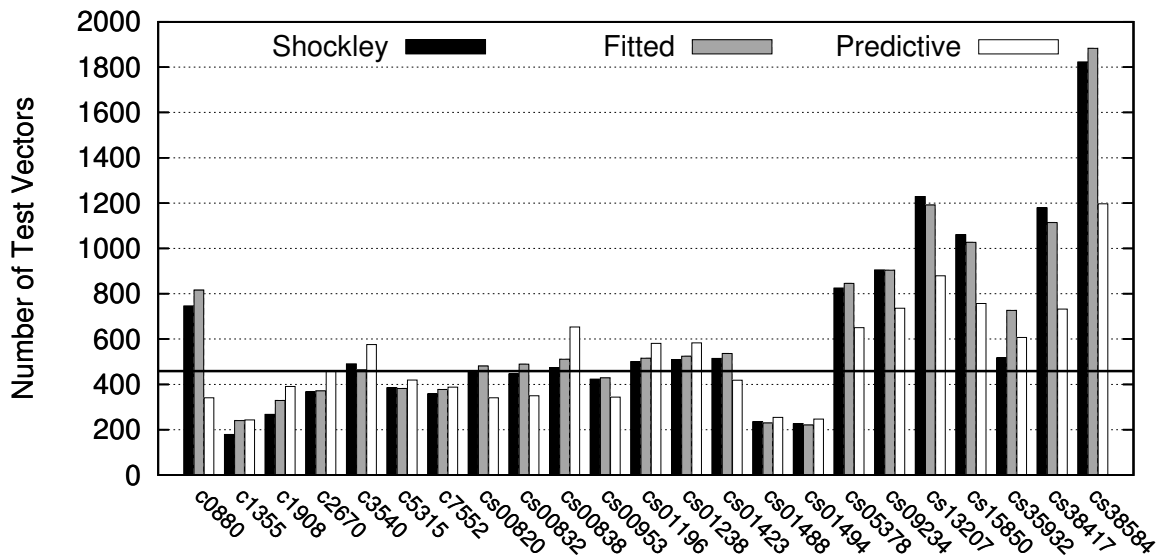


Figure 8.6: Number of test vectors for different technology models. Horizontal line indicates average over all circuits/technology models considered in Table 8.8.

Shockley model parameters were again taken from the model card of the AMS $0.35\ \mu\text{m}$ technology. We applied RBF-ATPG to ISCAS 85 and the combinational cores of the ISCAS 89 benchmark circuits. The fault list contained the same 10,000 randomly selected non-feedback bridging faults used for the aforementioned experiments with RBF-ATPG. All experiments were performed on a 2 GHz Pentium IV with 2 GB RAM.

Table 8.8 lists the results. The circuit name can be found in column one. Columns two and three repeat the number of faults and the number of test vectors obtained for the Shockley model (from Table 8.4, columns two and four). Subsequent columns four and five give the number of test vectors obtained by RBF-ATPG for the Fitted and the Predictive model, respectively. In the last row of the table, average numbers can be found.

We can see that the number of generated vectors varies quite substantially with no clear trend. Yet, on average the lowest relative difference in the number of test vectors is observed when comparing the Shockley with the Fitted model. This might be explainable by the similarities of the two technologies which might result in comparable detection conditions. Overall there is, however, no indication that any of these models stands out in terms of ATPG complexity. This is also illustrated by Figure 8.6 which depicts the number of test vectors generated for a selection of circuits. The horizontal line indicates the average number of vectors (458.94) obtained for all circuits and all technology models.

To determine the impact of design shrinking on the quality of the test set, we simulated the test vectors generated for the Shockley model under the Fitted and the Predictive model using our simulator [J3].⁹ This may yield information on whether vectors generated

⁹The same results would be observable when using our simulator SUPERB from Chapter 7 instead.

for a design are still effective after porting the design to a more recent technology node. Fault efficacy G -FE was calculated using Equation (5.4.7) and the probability distribution ρ based on findings from [178]. This probability distribution particularly suits modern copper interconnect technology. Results for the Fitted and the Predictive model can be found in Table 8.8, columns six and seven, respectively. Note that the fault coverage of the simulated test vectors is always 100% under the Shockley model.

Our results indicate that test vectors generated for the Shockley model are still effective for the Fitted and the Predictive model. In particular, for the former model, average fault efficacy remains extremely high at 99.79% (standard deviation is 0.2). The lowest figure for an individual circuit, obtained for cs38584, is 99.14%. For the Predictive model test vectors, efficacy is less superior; on average 98.35% were calculated. The lowest coverage of 90.97% was observed for cs00386. Yet, as the standard deviation for all circuits is 1.66, this result seems to be an outlier. Taking into account the huge differences between the behavior of transistors modeled by Shockley's equations as opposed to transistor behavior under the BPTM/BSIM4 deep submicron model for 65 nm, this appears to be acceptable. Nevertheless, the results strongly advocate a fault simulation run after a major design shrinking to ensure appropriate resistive bridging fault coverage.

8.3 Conclusions

In this chapter we introduced RBF-ATPG, our automatic test pattern generator for non-feedback resistive bridging faults. In contrast to similar test pattern generators for resistive bridging faults, RBF-ATPG is able to accurately determine G -ADI, which is a prerequisite for the computation of the exact fault coverage metric G -FC. This is enabled by a refined algorithm which can cope with many non-trivial detection phenomena, including faults for which more than one test vector is required to completely cover their G -ADI. Our experimental results demonstrate that the efficiency of RBF-ATPG is higher than that of state-of-the-art ATPG tools for single-stuck-at faults.

Extensive experiments investigated the performance of test vectors generated for common fault models, such as the single-stuck-at model, in the detection of resistive shorts. Our results underline that dedicated test pattern generation for resistive shorts is required to ensure complete coverage of the defects. This is also supported by our experiments exploring the detection capabilities of n -detection and 4-way model test vectors. Further experiments studying the impact of different technology models on the test pattern generation process were also performed. It turns out that none of the technologies investigated stands out in terms of ATPG complexity. Moreover, we found that a new ATPG run may be adequate after a major design shrinking to ensure constantly high fault coverages.

Table 8.4: Experimental results for RBF-ATPG.

Circuit	Faults	UF	Vec	Eff	US	$\frac{G-ADI}{[0, R_{max}]}$	Time [s]	
							SAT	Total
c0017	2	0	2	1.00	0	96.69	0.00	0.00
c0095	77	0	18	4.28	51	87.88	0.00	0.01
c0432	5,253	38	732	7.18	591	97.48	4.28	15.56
c0499	8,985	136	54	166.39	304	97.94	309.88	321.46
c0880	10,000	9	745	13.42	1,816	94.56	32.53	47.83
c1355	10,000	15	178	56.18	4,167	94.40	75.25	96.95
c1908	10,000	139	267	37.45	981	97.39	173.60	185.85
c2670	10,000	41	366	27.32	917	97.92	103.09	114.48
c3540	10,000	59	489	20.45	1,323	97.34	3,017.08	3,051.61
c5315	10,000	6	384	26.04	480	99.10	622.41	641.63
c7552	10,000	10	357	28.01	704	98.78	1,554.13	1,579.12
cs00208	3,986	47	124	32.15	1,375	90.84	0.28	1.20
cs00298	4,468	34	125	35.74	817	94.73	0.52	1.61
cs00344	7,760	83	206	37.67	3,199	86.28	1.30	5.66
cs00349	7,881	79	197	40.01	3,374	85.92	1.46	5.88
cs00382	7,809	22	255	30.62	1,324	96.23	1.38	4.78
cs00386	9,384	110	77	121.87	2,077	93.97	0.86	3.09
cs00400	8,290	24	245	33.84	1,493	96.14	1.35	4.95
cs00420	10,000	72	317	31.55	3,398	91.56	2.32	7.96
cs00444	10,000	41	312	32.05	2,752	94.88	2.74	9.18
cs00510	10,000	32	320	31.25	5,122	88.09	1.49	9.71
cs00526	10,000	76	372	26.88	2,060	95.12	5.47	12.36
cs00641	10,000	21	304	32.89	272	99.21	1.94	4.61
cs00713	10,000	40	332	30.12	1,581	97.30	15.86	23.07
cs00820	10,000	138	454	22.03	5,302	87.99	5.46	18.70
cs00832	10,000	122	445	22.47	5,349	87.83	5.08	18.75
cs00838	10,000	65	473	21.14	3,490	91.57	11.06	21.36
cs00953	10,000	15	422	23.70	6,949	85.58	19.09	36.29
cs01196	10,000	43	500	20.00	2,301	95.08	55.08	66.70
cs01238	10,000	62	508	19.69	2,539	94.60	79.52	92.25
cs01423	10,000	62	514	19.46	2,099	93.06	28.78	43.51
cs01488	10,000	44	234	42.74	1,311	96.37	3.62	9.52
cs01494	10,000	42	225	44.44	1,387	96.16	3.78	10.47
cs05378	10,000	9	824	12.14	668	98.28	217.66	245.57
cs09234	10,000	50	904	11.06	941	97.41	1,479.33	1,541.57
cs13207	10,000	9	1,228	8.14	353	99.33	2,126.42	2,240.41
cs15850	10,000	8	1,060	9.43	501	98.99	4,576.99	4,684.61
cs35932	10,000	148	516	19.38	3,213	94.32	100,750.51	101,045.79
cs38417	10,000	1	1,178	8.49	1,678	94.16	51,801.47	52,233.31
cs38584	10,000	93	1,822	5.49	1,147	96.69	88,748.90	89,227.03
∅		51.13	452.13	30.35	1,985.15	94.43	6,396.05	6,442.11

Table 8.5: Performance of single-stuck-at test sets.

Circuit	Number of Vectors			G -FE (Stuck-at)
	RBF	Stuck-at	Top-up	
c0017	2	5	2	36.44
c0095	18	12	8	93.88
c0432	732	61	319	95.09
c0499	54	63	15	99.89
c0880	745	64	558	93.88
c1355	178	95	71	99.76
c1908	267	148	166	99.13
c2670	366	109	240	98.46
c3540	489	166	325	98.06
c5315	384	127	144	99.37
c7552	357	184	171	99.47
cs00208	124	36	76	96.18
cs00298	125	33	79	96.90
cs00344	206	24	127	96.06
cs00349	197	26	116	96.94
cs00382	255	34	162	96.41
cs00386	77	79	34	98.21
cs00400	245	35	177	95.88
cs00420	317	81	193	97.18
cs00444	312	34	229	96.40
cs00510	320	69	191	97.37
cs00526	372	63	253	96.44
cs00641	304	43	149	98.12
cs00713	332	49	122	98.58
cs00820	454	124	265	96.95
cs00832	445	126	273	96.84
cs00838	473	162	276	97.58
cs00953	422	102	293	97.46
cs01196	500	174	318	97.44
cs01238	508	178	329	97.49
cs01423	514	64	300	96.69
cs01488	234	134	113	99.11
cs01494	225	138	93	99.25
cs05378	824	155	424	97.79
cs09234	904	219	365	98.03
cs13207	1,228	307	346	98.76
cs15850	1,060	197	218	99.07
cs35932	516	56	129	98.75
cs38417	1,178	194	370	98.80
cs38584	1,822	209	487	97.72
\emptyset	452.13	104.48	213.15	96.05

Table 8.6: Efficiency comparison with Resistive Bridging Fault Test Generator RBFTG [107] and single-stuck-at ATPGs ATOM [62] and SPIRIT [54].

Circuit	Eff	RBFTG [107]			#s@- faults	ATOM [62]		SPIRIT [54]	
		Faults	Vec	Eff		Vec	Eff	Vec	Eff
c0017	1.00	n/a	n/a	n/a	22	n/a	n/a	9	2.44
c0095	4.28	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
c0432	7.18	n/a	n/a	n/a	524	110	4.76	72	7.28
c0499	166.39	n/a	n/a	n/a	758	127	5.97	98	7.73
c0880	13.42	n/a	n/a	n/a	942	133	7.08	78	12.08
c1355	56.18	n/a	n/a	n/a	1,574	192	8.20	155	10.15
c1908	37.45	n/a	n/a	n/a	1,879	210	8.95	185	10.16
c2670	27.32	n/a	n/a	n/a	2,747	242	11.35	194	14.16
c3540	20.45	n/a	n/a	n/a	3,428	264	12.98	242	14.17
c5315	26.04	n/a	n/a	n/a	5,350	216	24.77	215	24.88
c7552	28.01	n/a	n/a	n/a	7,550	393	19.21	318	23.74
cs00208	32.15	231	131	1.76	217	65	3.34	42	5.17
cs00298	35.74	784	274	2.86	308	52	5.92	48	6.42
cs00344	37.67	815	159	5.13	342	62	5.52	29	11.79
cs00349	40.01	n/a	n/a	n/a	350	65	5.38	33	10.61
cs00382	30.62	770	348	2.21	399	72	5.54	46	8.67
cs00386	121.87	877	355	2.47	384	109	3.52	82	4.68
cs00400	33.84	487	251	1.94	424	71	5.97	40	10.60
cs00420	31.55	643	245	2.62	455	98	4.64	65	7.00
cs00444	32.05	703	273	2.58	474	77	6.16	45	10.53
cs00510	31.25	916	241	3.80	564	90	6.27	n/a	n/a
cs00526	26.88	873	579	1.51	555	107	5.19	82	6.77
cs00641	32.89	688	405	1.70	467	99	4.72	n/a	n/a
cs00713	30.12	708	330	2.15	581	100	5.81	88	6.60
cs00820	22.03	964	694	1.39	850	190	4.47	n/a	n/a
cs00832	22.47	965	681	1.42	870	200	4.35	169	5.15
cs00838	21.14	836	254	3.29	857	183	4.68	n/a	n/a
cs00953	23.70	906	328	2.76	1,079	138	7.82	n/a	n/a
cs01196	20.00	871	455	1.91	1,242	227	5.47	216	5.75
cs01238	19.69	911	448	2.03	1,355	240	5.65	229	5.92
cs01423	19.46	779	353	2.21	1,515	135	11.22	95	15.95
cs01488	42.74	963	430	2.24	1,486	196	7.58	176	8.44
cs01494	44.44	957	464	2.06	1,506	191	7.88	178	8.46
cs05378	12.14	977	946	1.03	4,603	358	12.86	386	11.92
cs09234	11.06	983	663	1.48	6,927	660	10.50	633	10.94
cs13207	8.14	979	430	2.28	9,815	709	13.84	710	13.82
cs15850	9.43	954	601	1.59	11,725	643	18.23	675	17.37
cs35932	19.38	998	122	8.18	39,094	129	303.05	373	104.81
cs38417	8.49	n/a	n/a	n/a	31,180	1,458	21.39	1,585	19.67
cs38584	5.49	n/a	n/a	n/a	36,303	989	36.71	1,419	25.58

Table 8.7: Test vector sets for n -detection and 4-way model and their resistive bridging fault coverage.

Circuit	1-detection			5-detection			10-detection			4-way		
	EVec	Vec	G-FE	EVec	Vec	G-FE	EVec	Vec	G-FE	EVec	Vec	G-FE
c5315	85	85	99.03	197	223	99.82	253	418	99.93	236	357	99.99
c6288	24	24	97.19	56	56	99.42	90	94	99.81	83	85	99.52
c7552	94	94	99.02	257	374	99.91	297	735	99.97	267	384	99.95
cs15850	124	129	98.41	319	496	99.73	372	969	99.92	254	336	99.94
cs38417	119	119	98.13	335	507	99.55	399	832	99.68	224	241	99.58
cs38584	130	130	96.80	435	614	99.12	528	1147	99.51	232	266	99.28
\emptyset, Σ	576	581	98.10	1,599	2270	99.59	1939	4195	99.80	1296	1669	99.71

Table 8.8: Comparison of ATPG results for different technology models.

Circuit	Number of Faults	Number of Patterns Generated for Model			Fault Efficacy G -FE (Shockley Vectors)	
		Shockley	Fitted	Predictive	Fitted	Predictive
c0095	77	18	20	16	99.20	98.74
c0432	5,253	732	659	489	99.83	99.71
c0880	10,000	745	816	341	99.87	99.73
c1355	10,000	178	240	243	99.70	94.83
c1908	10,000	267	329	391	99.73	98.24
c2670	10,000	366	371	459	99.78	98.59
c3540	10,000	489	464	576	99.84	98.05
c5315	10,000	384	382	419	99.86	99.34
c7552	10,000	357	377	388	99.81	99.56
cs00208	3,986	124	120	145	99.97	97.11
cs00298	4,468	125	158	141	99.54	99.06
cs00344	7,760	206	221	145	99.93	97.99
cs00349	7,881	197	172	149	99.98	97.91
cs00382	7,809	255	253	198	99.91	99.56
cs00386	9,384	77	72	164	99.99	90.97
cs00400	8,290	245	276	205	99.89	99.58
cs00420	10,000	317	305	405	99.96	96.90
cs00444	10,000	312	320	238	99.81	99.62
cs00510	10,000	320	334	263	99.87	99.69
cs00526	10,000	372	432	264	99.75	99.52
cs00641	10,000	304	333	352	99.95	98.60
cs00713	10,000	332	364	324	99.93	98.55
cs00820	10,000	454	481	341	99.79	99.22
cs00832	10,000	445	489	350	99.75	98.96
cs00838	10,000	473	511	653	99.92	96.91
cs00953	10,000	422	429	344	99.94	99.65
cs01196	10,000	500	515	581	99.77	96.82
cs01238	10,000	508	524	583	99.75	96.87
cs01423	10,000	514	536	418	99.92	99.60
cs01488	10,000	234	230	254	99.96	98.03
cs01494	10,000	225	221	247	99.96	97.77
cs05378	10,000	824	846	650	99.95	99.56
cs09234	10,000	904	904	736	99.72	98.30
cs13207	10,000	1,228	1,192	879	99.60	98.98
cs15850	10,000	1,060	1,027	757	99.73	99.10
cs35932	10,000	516	726	607	99.38	98.52
cs38417	10,000	1,178	1,114	732	99.75	99.52
cs38584	10,000	1,822	1,883	1,197	99.14	97.68
∅		474.45	490.68	411.68	99.79	98.35

9 Advanced Testing Methods

This chapter addresses testing methodologies which are meant to extend or even complement the “traditional” static voltage test in order to increase the coverage of resistive shorts. In this respect they may be understood as advanced or non-standard testing methods. The resistive bridging fault model was originally aimed at the “traditional” testing approach. Yet, due to the flexibility of its analytical concept, it is possible to extend the model such that it reflects the impact of advanced methods on the detection of resistive shorts as well. In the following, we will describe both the necessary extensions as well as the coverage impact of the advanced methods.

Two conceptually different approaches will be investigated. The first approach aims at modifying the operating conditions the device under test is exposed to during test application. We will consider the effect of controlling both voltage and temperature, i.e. low-voltage and low-temperature testing, on the coverage of shorts. The second strategy, known as quiescent current, or (Delta-)IDDQ testing, exploits the fact that a resistive short may increase the amount of current drawn by a circuit. This increase can be measured and used as an indicator for the presence of a defect. Both approaches are particularly effective in detecting *hard defects*. Hard defects cause failures under regular operating conditions and may be detected by “traditional” voltage testing. In addition to that, each of the two approaches has been reported to be successful in the detection of *flaws* [63] in the so-called weak ICs. These are defects resulting in infant mortality¹, reduced reliability, or transient faults, rather than catastrophic failures. Yet, flaws may become catastrophic in the future, due to aging processes such as gate oxide breakdown, hot carrier effects and electromigration [133, 204]. There are alternative methods to detect flaws, such as burn-in [76], which are, however, often associated with considerable costs.

We will show that the extensions to the resistive bridging fault model allow us to determine the efficacy of both approaches on the detection of hard (short) defects. Furthermore, we will also introduce metrics which make it possible to estimate the coverage of flaws. Low-voltage and low-temperature testing are explored in Chapter 9.1. Chapter 9.2 demonstrates that (Delta-)IDDQ testing can be integrated into the resistive bridging fault model which allows to evaluate the effectiveness of voltage and current testing within the same framework. Conclusions with respect to both techniques are given in Chapter 9.3.

¹If a product fails during the phase of its “initial” use by the customer this is referred to as “infant mortality”; this is also called an “early life failure”. The extent of the “initial” phase depends on the reliability requirements for the product under consideration.

9.1 Low-Voltage and Low-Temperature Testing

It is well-known that testing at reduced power supply voltage, i.e. *low-voltage* testing, as well as testing at reduced operating temperature, also called *low-temperature* testing, enables the detection of defects which have escaped conventional test runs. Numerous studies have demonstrated the effectiveness of low-voltage testing. Hao et al. [63] reported on experiments with manufactured ICs, Renovell et al. [156] performed a mathematical analysis for resistive bridging faults. For the same class of faults Liao et al. [103] contributed data on electrical simulation with SPICE and Sar-Dessai et al. [165] provided results from a fault simulation experiment. Kruseman et al. [93, 95] compared the effectiveness of IDDQ and low-voltage testing. Note that some authors refer to low-voltage testing as very-low-voltage testing. A related technique is called MinVDD testing [14, 188]. Low-temperature testing was investigated at Intel corporation (United States of America) where it has been deemed to be effective in detecting three defect classes observed in manufactured devices [128]. In [189] low-temperature testing was found to be an efficient supplement to burn-in and helps to pinpoint defects in resistive silicide (i.e. TiSi_2 used to enhance the conductivity of polycrystalline silicon). It has also been applied successfully in combination with IDDQ testing [82].

Low-temperature testing is a cost-intensive technique as it requires special equipment, such as a thermal chuck to control the temperature of the device during test application. In contrast to, that low-voltage testing involves no additional costs. However, as a reduction in the power supply voltage degrades the operating speed of the device, the frequency at which the test vectors are applied has to be reduced [192]. This implies that the overall test time is increased. Or, if this is not acceptable, the test set has to be truncated appropriately. Theoretically, the minimal power supply voltage at which a conventional CMOS logic gate will function correctly, is slightly above the maximum absolute threshold voltage of any transistor (denoted by V_{tn0} and V_{tp0} for n - and p -transistors, respectively) within that gate [63]. Yet, in practice a whole IC will not be able to operate properly at the theoretical minimal voltage, due to e.g. noise issues. The question which value of V_{dd} is optimal for low-voltage testing is, for instance, explored in [26, 93]. However, no clear consensus is found: While [93] opts for $1.5 \cdot V_T$, with V_T being defined as $V_T = \min(V_{tn0}, |V_{tp0}|)$, the authors of [26] see an optimal range between $2.0 \cdot V_T$ and $2.5 \cdot V_T$.

To simplify the discussion we will use the term *nominal conditions* to denote the operating conditions, the device is exposed to during normal operation. Testing under *non-nominal conditions* and *low-X* testing refers to either testing under reduced temperature, reduced voltage, or both.

If for a given test set a short with a certain resistance is detected by low-X testing but not by testing under the nominal conditions, this may have two reasons:

1. Theoretically the defect is detectable under the nominal conditions, i.e. it is a hard defect. Yet, none of the test vectors which allows for the defect's detection is contained in the test set given.

2. There exists no test vector which detects the defect under the nominal conditions (e.g. because its intrinsic resistance is too high). In this case the defect is deemed to be a flaw and is commonly considered as redundant.

In the first case, low-X testing improved the detection capabilities of the test set. A defect from the second category might be considered irrelevant. Yet, as already mentioned, various aging mechanisms are likely to aggravate this kind of defect. Consequently, the defective IC might fail burn-in or become an early life failure. Depending on the application these weak ICs might be intolerable.

In the following we will investigate the effect of low-voltage testing, low-temperature testing and their combination on the detection of resistive bridging faults – a class of defects which contributes many flaws. This study has been adapted from our publications [W5], [P6, P9], and [J5].² We will extend the resistive bridging fault model to be able to account for the impact of power supply voltage and temperature on transistor behavior. Furthermore, we will take the temperature dependence of the short's resistance into consideration. This will enable us to exactly quantify the effectiveness of both techniques for the detection of resistive bridging faults, while at the same time accurately discriminating between the coverage of flaws and that of hard defects. We will not consider the impact of statistical process variations on the detection of defects.

We will consider two application scenarios:

Scenario CN (cost neutral) assumes that additional costs due to low-X testing are unacceptable. This implies that low-temperature testing, which is very cost-intensive, may not be used and low-X testing is restricted to low-voltage testing only. Testing at reduced voltage, however, degrades the operating speed of the device under test and, thus prolongs test application time. Since this implies increased cost we have to truncate the test set to keep test time and cost constant. Scenario **CN** is designed to reflect the requirements of high-volume manufacturing test.

Scenario AS (additional screening) targets products with elevated reliability requirements. This means that an increase of test costs is tolerable if test quality is improved. Under Scenario **AS** we allow for two test runs, one under the nominal conditions followed by a low-X test application run. Both low-voltage and low-temperature testing may be employed. The complete test set is applied irrespective of a potential increase in test time.

To quantify the efficiency of low-X testing for the detection of resistive shorts, we introduce several new fault coverage metrics. They are designed to accurately discriminate between the coverage of flaws and that of hard defects. Furthermore, we account for the impact of performance degradation in Scenario **CN**. We integrated the modified resistive bridging fault model and the coverage metrics into the resistive bridging fault simulator from [J3]. Numerous experiments were performed with this tool. Results enable us to decide whether it is more beneficial under Scenario **CN** to perform low-voltage testing with a truncated

²We would like to thank Prof. I. Pomeranz (Purdue University, USA) for her contributions to this work.

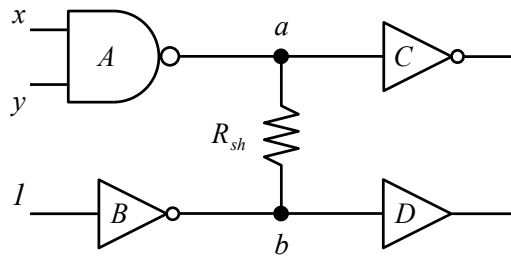


Figure 9.1: Example of circuit with a resistive short affecting nodes a and b .

test set or apply the unabbreviated test set at the nominal power supply voltage. Under both Scenario **CN** and **AS** we are able to suggest for each circuit values for temperature and power supply voltage which yield the best coverage. Moreover, we describe a situation in which low-voltage testing, contrary to conventional wisdom, actually results in coverage loss and quantify its extent.

The extensions and modifications of the resistive bridging fault model which allow to address the impact of low-X testing are described in Chapter 9.1.1. Subsequently we present experimental results obtained with the modified simulator from [J3] in Chapter 9.1.2. An example for coverage loss is given in Chapter 9.1.3. There we also introduce metrics quantifying this effect and present experimental results. Conclusions are given in Chapter 9.3.

9.1.1 Extensions of the Resistive Bridging Fault Model

Consider the circuit from Figure 9.1 in which a defect shorts nodes a and b driven by gates A and B , respectively. To highlight the impact of low-X testing rather than reiterating the propagation phenomena of the RBF model, we assume two simplifications: (1) all possible activating assignments induce the logical value 1 at the input of gate B (resulting in the logical value 0 at node b), and (2) fault effects propagated via gate D can be ignored when evaluating the defect detection (e.g. in a larger circuit propagation of fault effects might be blocked for the activating assignments). Consequently, only assignments to inputs x and y of gate A , which induce the logical value 1 at the gate's output, activate the bridging fault.

Assume we apply the pattern $(0, 0, 1)$ at the nominal V_{dd} to the inputs of gates A and B .³ Due to the input assignment both p -transistors in the pull-up network of A are activated. In gate B the pull-down network is active. For this scenario, the characteristic of voltage V_{001}^{nom} at node a is illustrated in Figure 9.2 (we omitted the characteristic at node b to increase readability). In the nominal case, the logic threshold of the input of gate C is $V_{lt}^{C,nom}$. The projection of the intersection between this threshold and voltage characteristic

³We assume circuits with one single nominal power supply voltage. Circuits that employ adaptive power management techniques such as dynamic voltage scaling (DVS) have to operate correctly under multiple supply voltages. Our framework can be easily extended to such circuits using strategies similar to the ones discussed in [85].

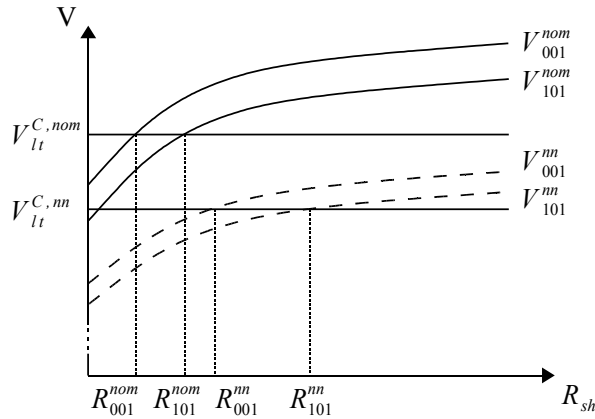


Figure 9.2: Voltage characteristic at node a as a function of R_{sh} for nominal (“nom”) and non-nominal (“nn”) operating conditions and two test vectors.

V_{001}^{nom} to the x -axis yields the critical resistance R_{001}^{nom} . Therefore, the faulty logical value 0 is interpreted by gate C for any short having a resistance in the range $[0, R_{001}^{nom}]$, which is also the ADI for this pattern.

When we apply the input pattern $(1, 0, 1)$ to the driving gates’ inputs, only a single p -transistor in the pull-up network network of A is activated. This implies that the voltage characteristic V_{101}^{nom} at node a is shifted and we obtain a different critical resistance R_{101}^{nom} for this pattern. Fault effects are thus interpreted by gate C for any short having a resistance in the range $[0, R_{101}^{nom}]$. This is also the ADI for this pattern which in addition contains the ADI for pattern $(0, 0, 1)$. Due to the symmetric transistor structure of the NAND gate, input pattern $(0, 1, 1)$ leads to the same situation and thus $R_{011}^{nom} = R_{101}^{nom}$ holds. Input pattern $(1, 1, 1)$ does not activate the defect and consequently yields an empty ADI. Thus, C -ADI for test pattern set $\{(0, 0, 1), (1, 1, 1)\}$ is $[0, R_{001}^{nom}] \cup \emptyset = [0, R_{001}^{nom}]$ and G -ADI for this bridging fault is $[0, R_{101}^{nom}]$.

So far we have assumed that testing is performed under the normal operating conditions of the device. However, as already mentioned, it can be beneficial to modify the operating conditions, namely temperature and/or power supply voltage, during test application. This will in general affect the voltage characteristics at the shorted nodes, lead to different critical resistances, and consequently result in other ADIs than those obtained when testing under the nominal operating conditions. We denote C -ADI and G -ADI derived for the nominal conditions as C^{nom} and G^{nom} , respectively. Their equivalents for the non-nominal conditions are referred to as C^{nn} and G^{nn} , respectively. To compute C^{nn} and G^{nn} we may use the same procedures valid for the nominal case. Yet, the electrical equations founding the basis of the resistive bridging fault model have to be modified accordingly. Before we describe these modifications in more detail, we will analyze the influence of non-nominal operating conditions within the bounds of the model and propose metrics to assess their coverage impact. Subsequently, means to account for the performance degradation involved with low-voltage testing are developed. Finally, we will cover the changes imposed on the electrical model.

Metrics for Detection Under Non-Nominal Conditions

Refer again to Figure 9.2 and assume we reduce the power supply voltage V_{dd} below its nominal value. In this case the voltage characteristics at node a for both input patterns $(0, 0, 1)$ and $(1, 0, 1)$ will be shifted towards lower voltage levels. Moreover, the logic threshold of gate C , which is also dependent on V_{dd} , will be scaled as well. Possible voltage characteristics for this scenario are indicated as V_{001}^{nn} and V_{101}^{nn} , respectively. The reduced logic threshold is denoted as $V_{lt}^{C,nn}$. As can be seen, new critical resistances R_{001}^{nn} and R_{101}^{nn} are obtained for this case. Thus, C^{nn} of the test set $\{(0, 0, 1), (1, 1, 1)\}$ is $[0, R_{001}^{nn}]$ and G^{nn} is $[0, R_{101}^{nn}]$.

By reducing V_{dd} , both C -ADI and G -ADI have been enlarged beyond the range observed under the nominal conditions – this coincides with the results obtained in [63, 103, 156]. In particular, it holds that $C^{nom} \subset C^{nn}$, i.e. the range of resistances detected under the non-nominal conditions exceeds that obtained by testing under the nominal conditions while keeping the test set unchanged. Furthermore, $G^{nom} \subset C^{nn}$ holds which means that in the non-nominal case even the suboptimal test set manages to detect the defect for any short resistance R_{sh} , which is detectable under the nominal conditions. In addition to that, it detects some flaws having a resistance from the interval $[R_{101}^{nom}, R_{001}^{nn}]$. In summary, testing under the non-nominal conditions has two effects on the coverage of defects:

1. It may increase the coverage of hard defects, which are shorts detectable at the nominal conditions, i.e. shorts for which $R_{sh} \in G^{nom}$ holds.
2. Non-nominal testing may contribute flaw coverage, i.e. detect some defects which are undetectable by static (e.g. scan) testing under the nominal conditions irrespective of the test set used. The resistance R_{sh} of a flaw is from the range $[0 \Omega, \infty] \setminus G^{nom}$.

The second category includes defects that are detectable by delay testing only, latent defects which might deteriorate and lead to early life failures [133], and defects which are redundant in terms of the resistive bridging fault model (refer again to Chapter 5.3). Metrics assessing the impact of non-nominal testing have to correctly discriminate between hard defects and flaws.

The coverage of hard defects by low-X testing is evaluated by our *non-nominal fault coverage* metric. It accounts for the range of short resistances detected by non-nominal testing which is detectable under the nominal conditions as well ($R_{sh} \in C^{nn} \cap G^{nom}$). This interval is related to the (non-empty) range of resistances detectable under the nominal conditions (Figure 9.3 illustrates the metric in form of a Venn diagram). Our nominal fault coverage computes as:

$$FC^{nn} = 100\% \cdot \frac{\int_{(C^{nn} \cap G^{nom})} \rho(r) dr}{\int_{G^{nom}} \rho(r) dr}. \quad (9.1.1)$$

Metric FC^{nn} may be seen as an extension of G -FC (see Equation (5.4.4)) capturing the coverage of hard defects when performing low-X testing only (Scenario **CN**).⁴

⁴In fact for $C^{nn} = C^{nom}$ both metrics FC^{nn} and G -FC are equivalent (recall that $C^{nom} \subset G^{nom}$).

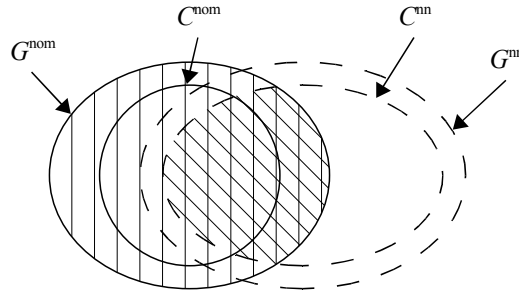


Figure 9.3: Venn diagram for non-nominal fault coverage FC^{nn} (Equation (9.1.1)). Diagonal lines indicate the numerator, vertical lines show the denominator.

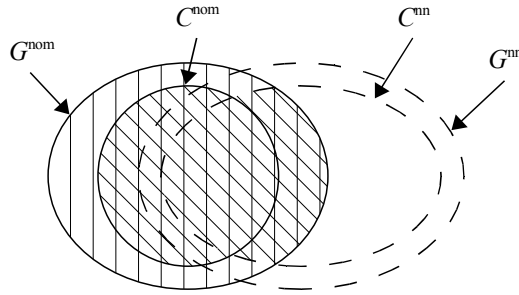


Figure 9.4: Venn diagram for combined fault coverage FC_{comb}^{nn} (Equation (9.1.2)). Diagonal lines indicate the numerator, vertical lines show the denominator.

In Scenario **AS** we allow for two test runs, one under the nominal and one under the non-nominal conditions. We developed the *combined fault coverage* metric to evaluate the probability that a hard defect is detected during either of the two test runs. In this case the defect has a resistance $R_{\text{sh}} \in (C^{\text{nom}} \cup C^{\text{nn}}) \cap G^{\text{nom}}$. The metric is visualized in Figure 9.4 and is defined as follows:

$$FC_{\text{comb}}^{\text{nn}} = 100\% \cdot \frac{\int_{((C^{\text{nom}} \cup C^{\text{nn}}) \cap G^{\text{nom}})} \rho(r) dr}{\int_{G^{\text{nom}}} \rho(r) dr}. \quad (9.1.2)$$

Both FC^{nn} and $FC_{\text{comb}}^{\text{nn}}$ focus on the detection of hard defects. To capture the coverage of flaws, however, we propose the *flaw coverage* metric. Flaws are undetectable at the nominal conditions ($R_{\text{sh}} \in [0, \infty) \setminus G^{\text{nom}}$) but are detected by low-X testing ($R_{\text{sh}} \in C^{\text{nn}}$). Consequently, flaws detected by non-nominal testing must have a resistance $R_{\text{sh}} \in ([0, \infty) \setminus G^{\text{nom}}) \cap C^{\text{nn}}$. The metric relates this resistance range to the interval of resistances representing all flaws as follows (see Figure 9.5 for an illustration):

$$FC_{\text{flaw}}^{\text{nn}} = 100\% \cdot \frac{\int_{(([0, \infty) \setminus G^{\text{nom}}] \cap C^{\text{nn}})} \rho(r) dr}{\int_{[0, \infty) \setminus G^{\text{nom}}} \rho(r) dr}. \quad (9.1.3)$$

Assume for instance that C^{nom} is $[0, 800]$, G^{nom} is $[0, 1000]$, and that due to test application at the non-nominal conditions, C^{nn} becomes $[0, 1250]$ and G^{nn} extends to $[0, 1400]$. Hence, in the numerator of $FC_{\text{flaw}}^{\text{nn}}$, we obtain $[1000, 1250]$. This is the range of short resistances of

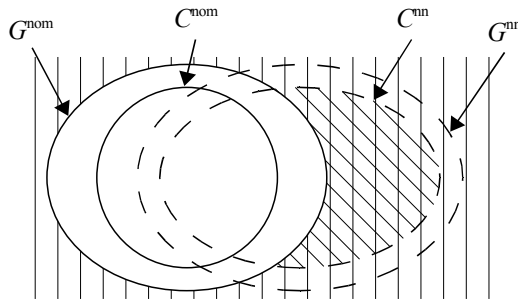


Figure 9.5: Venn diagram for flaw coverage $FC_{\text{flaw}}^{\text{nn}}$ (Equation (9.1.3)). Diagonal lines indicate the numerator, vertical lines show the denominator.

those defects which can be detected by non-nominal testing only. The denominator will be $[1000, \infty]$. Certainly the definition of this metric is only sound if there exists a limit R_{lim} such that $\rho(r) = 0$ for any $r > R_{\text{lim}}$. Yet, since for short defects the probability density function ρ is monotonic and decreasing [158], and the size distribution of particles that may cause short defects is also decreasing [180], we can definitely assume the existence of the limit R_{lim} .

Impact of Performance Degradation Implied by Low-Voltage Testing

When performing low-voltage testing the operating speed of the device under test is degraded with respect to the nominal operating conditions. In order to account for this degradation the frequency at which the test vectors are applied to the device has to be reduced. This in turn leads to an increased test application time. When prolonged test time (and the associated cost) is unacceptable one has to decide whether to apply the given test set TS “as it is” at the nominal power supply voltage, or to reduce V_{dd} and frequency and to apply a sub-set $TS' \subsetneq TS$ that preserves test application time.

This is exemplified in Figure 9.6 for test set $TS = \{v_0, \dots, v_5\}$ which contains six vectors. We define one *time unit* as the time required to apply one test vector, i.e. the duration of one clock cycle, at the circuit’s nominal operating frequency. Consequently the total test time is chosen such that the complete application of TS at this frequency is possible. If the power supply voltage is reduced, individual vectors may detect more defects but the time needed to apply a single vector increases, and, thus, exceeds one time unit. This means that less vectors can be applied if the total test application time is fixed (as in Scenario CN). In our example, test set TS has to be cut back to four vectors ($TS' = \{v_0, \dots, v_3\}$) to keep test time within the allowed budget.

Refer again to our example circuit from Figure 9.1. Assume that under the nominal conditions we apply test set $TS = \{(0, 0, 1), (1, 0, 1)\}$ while in the low-voltage testing scenario our restricted time budget allows for the application of test set $TS' = \{(0, 0, 1)\}$ only. We obtained TS' from TS by truncating the second test vector. Now we have to decide whether to perform the test under the nominal conditions, and apply the full test set TS (including vector $(1, 0, 1)$ which is more effective than $(0, 0, 1)$), or to reduce the

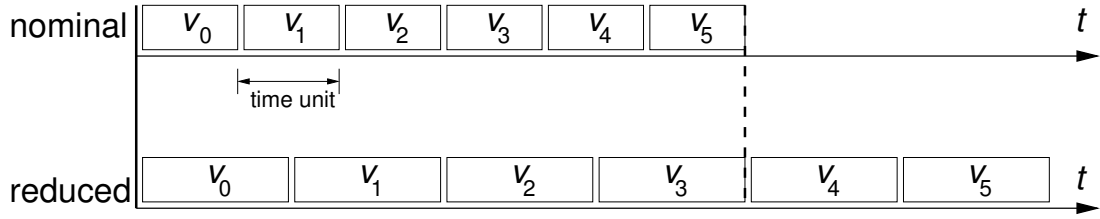


Figure 9.6: Performance degradation due to low-voltage testing.

power supply voltage, lower the frequency, and apply the truncated set TS' only. From our analysis we know that TS detects all short resistances from $[0, R_{101}^{\text{nom}}]$ (which is also the G -ADI for nominal V_{dd}) and TS' detects $[0, R_{001}^{\text{nn}}]$ under the non-nominal conditions. Since $R_{101}^{\text{nom}} < R_{001}^{\text{nn}}$, it is obvious that the low-voltage testing scenario is superior as it covers G^{nom} and detects some additional flaws as well. However, if $R_{001}^{\text{nn}} < R_{101}^{\text{nom}}$ would hold, leaving $(1, 0, 1)$ in TS appeared to be the better choice than lowering the power supply voltage. In this simple example, it would actually be more efficient to apply only vector $(1, 0, 1)$ at low-voltage (instead of $(0, 0, 1)$). However, in more complex circuits, excluding vectors which are less effective for one bridging fault could result in loss of coverage of other defects for which the vectors have originally been generated.

Our analysis, following in Chapter 9.1.2, will give more insight into the trade-off between the vectors' increased defect detection capabilities caused by low-voltage testing and the test set truncation implied by the frequency reduction. In particular, we will be able to make recommendations regarding the testing conditions which lead to a better defect coverage. In order to do so, we have to be able to make concise assertions about the performance degradation caused by low-voltage testing. A time unit has been defined as the duration of one clock cycle at the nominal operating frequency. Consequently, when testing under the nominal conditions, k test vectors can be applied in k time units. Let the performance degradation implied by low-voltage testing be expressed by the factor $\tau_{\text{pd}} \in \mathbb{R}_{\geq 1}$ ($\tau_{\text{pd}} = 1.1$ would mean that circuit performance is reduced by 10%). Then, under the non-nominal conditions, $\lfloor k/\tau_{\text{pd}} \rfloor$ test vectors can be applied in the same period of time. The fault coverage impact of low-voltage testing is evaluated by first performing resistive bridging fault simulation of the test set $TS = \{t_1, t_2, \dots, t_m\}$ under the nominal conditions and computing fault coverage G -FC according to Equation (5.4.4). Then simulation is repeated under the non-nominal conditions for the truncated test set $TS' = \{t_1, t_2, \dots, t_{m'}\}$, where $m' = \lfloor m/\tau_{\text{pd}} \rfloor$. Subsequently FC^{nn} is calculated using Equation (9.1.1). Finally, comparing G -FC and FC^{nn} gives the answer whether testing under the nominal conditions or low-voltage testing detects more defects within an identical period of time and, thus, is more effective under the test time constraint imposed by Scenario CN.

Voltage- and Temperature-Dependence Model

Changes in the circuit's operating conditions which affect power supply voltage and/or temperature do not have any implications on the fault effect propagation. They do, however,

necessitate modifications of the electrical model used to determine critical resistances (refer again to Chapter 5.2).⁵ We have to account for two basic effects:

1. The resistance of the short defect itself, R_{sh} , is a function of temperature T .
2. A critical resistance R_{crit} obtained under the nominal operating conditions shifts to a new value $R_{\text{crit}}^{\text{nn}}$ due to a change in voltage and/or temperature. This affects both C -ADI and G -ADI.

In the following we will describe the modifications required to integrate these two effects into the electrical model of the resistive bridging fault model.

Dependence of R_{sh} on T : Changes in temperature affect the resistivity of a material. A reasonable approximation of this effect is given by (see e.g. [132, p. 81f]):

$$R = R_{\text{ref}} \cdot (1 + \alpha \cdot (T - T_{\text{ref}})), \quad (9.1.4)$$

where R_{ref} is the resistance at temperature T_{ref} , α is the *temperature coefficient of electrical resistance* at temperature T_{ref} (usually $293.15 \text{ K} = 20^\circ \text{C}$) and T is the actual temperature. For metals, resistance rises with increasing temperature. Temperature coefficients α of metals used in semiconductor processing range between 0.003715 K^{-1} and 0.005866 K^{-1} .

Dependence of R_{crit} on V_{dd} : Power supply voltage V_{dd} is part of the equations used to calculate the critical resistance (see e.g. Equations (5.2.6) and (5.2.7)). Furthermore, changes in power supply voltage affect the logic thresholds of the driven gates. They may be determined analytically or from electrical equations. Using the modified parameters, $R_{\text{crit}}^{\text{nn}}$ is calculated just like R_{crit} .

Dependence of R_{crit} on T : We considered the temperature dependence of the transistor threshold voltages V_{tn0} and V_{tp0} , of the mobility μ , and of the intrinsic carrier concentration n_i . We used the temperature dependence model from Berkeley Predictive Technology Model (BPTM, which is provided by the Device Group at UC Berkeley) [22] in connection with the Berkeley Short-channel IGFET Model 4 (BSIM4) in version BSIM4.4.0 released in March 2004 (discussed in [208]).⁶

We calculated critical resistances for low-temperature testing scenarios using Equations (5.2.6) and (5.2.7) with adapted values of V_{tn0} , V_{tp0} , and μ . When computing equivalent transistors using Equations (5.2.10) and (5.2.11), we addressed the changes in substrate potential Φ which relies on the temperature dependent parameter n_i . Furthermore, we considered that the resistance of a short is reduced by a decrease in temperature according to Equation (9.1.4). This has the following effect: Suppose that the ratio between the resistance at nominal (high) temperature and the resistance at non-nominal (low) temperature is $1.2 : 1$. Assume furthermore, that we we calculated $R_{\text{crit}}^{\text{nn}} = 1000 \Omega$. Then, any short having

⁵Note that all effects described in this chapter also have to be considered when obtaining critical resistances from electrical simulations. In general, this implies that all precomputed data will have to be collected for every parameter/voltage setting while only a simple change of parameters is required in our (extended) analytical approach.

⁶The updated version BSIM4.6.1, available since May 2007, might induce changes to the material presented in the following.

a resistance of up to $1000\ \Omega$ which causes faulty logical values at non-nominal temperature will have a resistance of up to $1200\ \Omega$ under the nominal conditions. Consequently, R_{crit}^{nn} must be multiplied by a factor of 1.2.

9.1.2 Experimental Results

We applied 1,000 random test vectors to ISCAS 85 [19] circuits and the combinational cores of the ISCAS 89 [18] benchmarks. The fault list contained 10,000 randomly selected non-feedback bridging faults, where available.⁷ Alternatively, we could have also used layout extracted bridging faults (refer to Chapter 3.3). We employed the Shockley technology model and parameters from the SPICE model card of a $0.35\ \mu\text{m}$ technology from austriamicrosystems AG (AMS) to determine the critical resistances. Probability density function ρ proposed in [165] has been employed when calculating fault coverages. However, all experiments in this chapter could be repeated with any other short resistance distribution. All simulations were performed using a modified version of the simulator from [J3] which ran on a 2 GHz Pentium IV with 2 GB RAM. Yet, as modifications discussed in Chapter 9.1.1 are restricted to the electrical model, the simulator SUPERB from Chapter 7 could be used as well. Whenever required, G -ADI has been calculated using RBF-ATPG introduced in Chapter 8.

We decided to exclude feedback bridging faults from consideration. As the discussion in Chapter 5.5.3 and the detailed analysis in [P3][J1] have demonstrated, this type of bridging fault displays highly complex behavior. Fault coverages can only be determined if assumptions on the sensitivity of the automatic test equipment are made. Consequently, feedback bridging faults would have primarily introduced another stochastic variable, which distracted from the focus of this analysis.

Scenario CN

In Scenario CN, additional costs due to test application under the non-nominal conditions are unacceptable. We assume that only low-voltage testing is performed and that no additional testing under the nominal conditions is allowed. Test time may not exceed the time granted under the nominal conditions, thus, test sets have to be truncated appropriately.

During test-application we held the temperature constantly at 300 K. For the AMS $0.35\ \mu\text{m}$ technology under consideration, nominal power supply voltage V_{dd}^{nom} is 3.3 V. We considered several low-voltage settings, namely 3.0 V, 2.8 V, 2.5 V, and 2.0 V. To obtain the number of vectors to cut off, we determined the performance degradation factors τ_{pd} . We computed τ_{pd} by first calculating individual gate delays under the considered voltages and then performed critical path analysis (see e.g. [170]). The values of τ_{pd} ranged between 1.08 and

⁷Note that unless otherwise specified the fault lists used in this chapter are equal to those employed in all other experiments discussed in Part II of this work.

1.11 (i.e., between 8% and 11%) for 3.0 V, between 1.14 and 1.20 for 2.8 V, between 1.27 and 1.39 for 2.5 V, and between 1.66 and 1.91 for 2.0 V.

Fault coverages for 10, 100, and 1,000 time units can be found in Tables 9.3, 9.4, and 9.5, respectively. One time unit corresponds to the duration of one clock cycle at the circuit's nominal operating frequency. Column one of the tables gives the name of the circuit. This is followed in column two by the fault efficacy G -FE (Equation (5.4.7)) obtained under the nominal conditions. Columns three to six of the tables state FC^{nn} according to Equation (9.1.1) for the low-voltage settings. All faults redundant under the nominal conditions (i.e. $R_{sh} \notin G^{nom}$) had to be excluded when calculating FC^{nn} ; this ensures comparability with G -FE. Bold font indicates the maximum coverage achieved for the available number of time units per circuit. The last row of each table gives average coverages.

Consider the circuit c1355 and 100 time units (Table 9.4). For 3.3 V, which is the nominal V_{dd} , 100 test vectors have been applied, resulting in a fault efficacy G -FE of 97.02%. For 3.0 V, the performance degradation factor τ_{pd} has been determined to be 1.11. Thus, only 90 test vectors out of originally 100 could be applied; however, the non-nominal fault coverage achieved in this case is 97.13%. Note that the detections of defects not detectable at 3.3 V are not accounted for. Consequently, if only 100 time units are at our disposal, it is better to lower V_{dd} and to apply the first 90 vectors of the test set than to apply all 100 vectors at nominal V_{dd} . For 2.8 V, 2.5 V, and 2.0 V the value of τ_{pd} is 1.20, 1.38, and 1.91, and the numbers of applied vectors are 83, 72, and 52, respectively. From the figures in Table 9.4 it can be seen that fault coverage is maximal for 2.8 V.

Based on this data, we can conclude that under Scenario **CN**, the benefits of low-voltage testing are higher if more test time is available. For 10 time units the maximum fault coverage for most of the circuits (25 out of 38) is obtained at the nominal V_{dd} . The optimal voltage tends to drop, if 100 or even 1,000 time units are at our disposal. On average for 100 time units, the optimal voltage seems to be 2.8 V (16 circuits). For 1,000 time units, however, maximum fault coverage is achieved at 2.5 V for most of the circuits (21 in number).

The graphs in Figures 9.7 and 9.8 show the progression of the fault coverage (adjusted for performance degradation) over time for circuit c2670 and all considered values of V_{dd} . The performance degradation factors were 1.10 at 3.0 V, 1.19 at 2.8 V, 1.37 at 2.5 V, and 1.89 at 2.0 V. In the first 100 time units (Figure 9.7) fault coverage for 2.5 V, 2.8 V, 3.0 V, and the nominal voltage (indicated by a solid line) are very close. By contrast, the non-nominal coverage at 2.0 V is distinctively lower. Between time units 100 and 1,000 (Figure 9.8), the picture changes and the coverages obtained at non-nominal voltages exceed those for the nominal V_{dd} . In the figure the y -axis is restricted to the range between 90% and 100%. Even for a voltage of 2.0 V, at which the lowest number of vectors can be applied, the coverage surpasses the figures obtained under the nominal conditions after slightly over 100 time units. The highest coverage is achieved at 2.5 V. These observations are consistent with the conclusions previously drawn.

Table 9.6 reports the results for flaw coverage metric FC_{flaw}^{nn} from Equation (9.1.3) and 1,000 time units. The structure of the table is similar to the one of Tables 9.3–9.5. The flaw

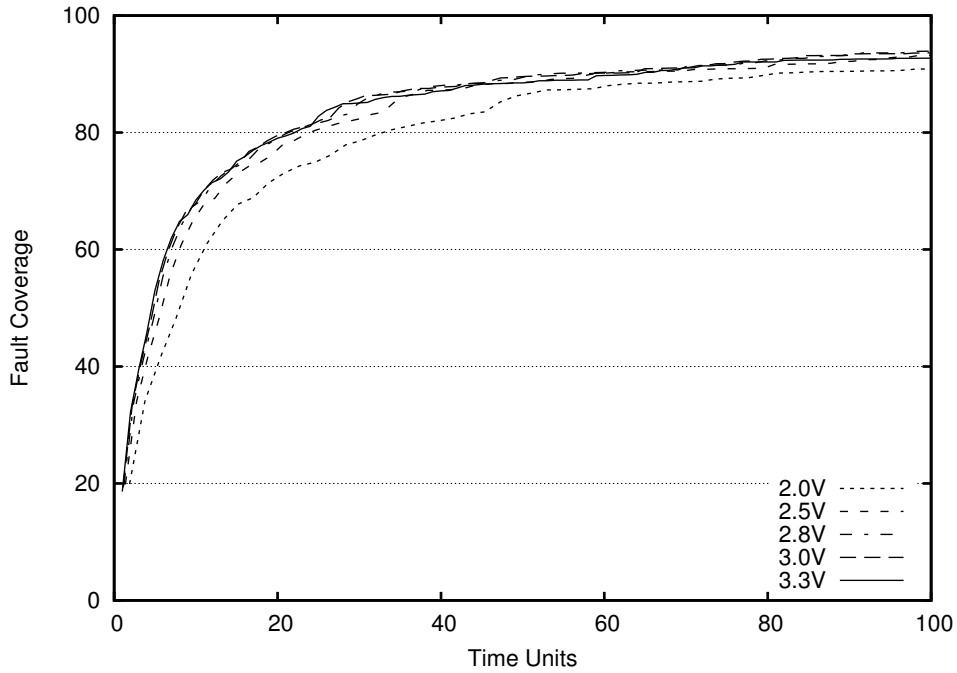


Figure 9.7: FC^{nn} for c2670 and different values of V_{dd} as function of test time for time units 0 through 100.

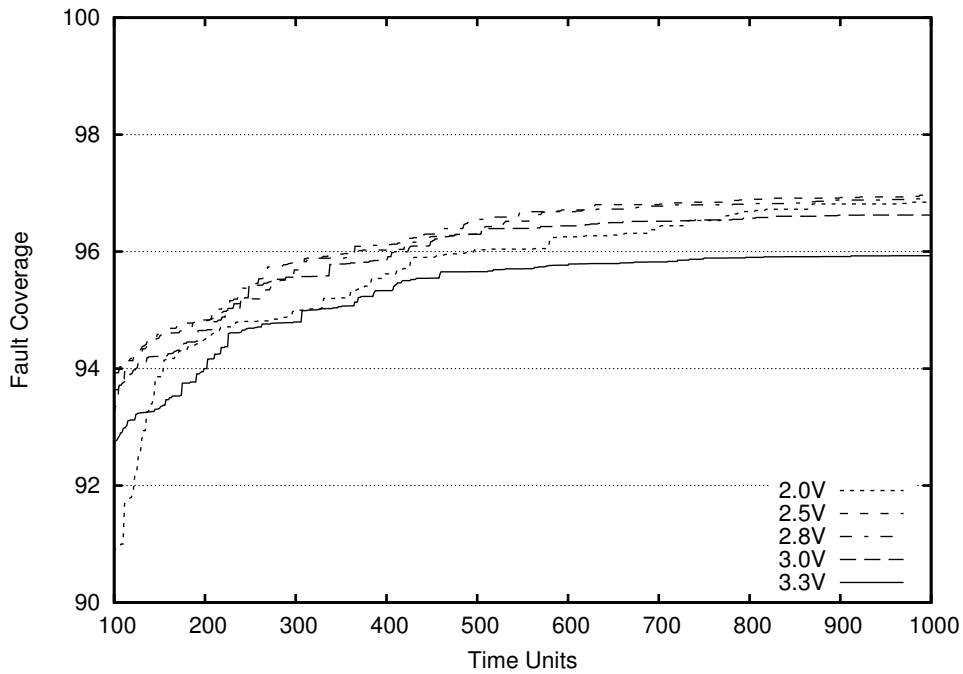


Figure 9.8: FC^{nn} for c2670 and different values of V_{dd} as function of test time for time units 100 through 1,000.

coverage at nominal V_{dd} has been excluded as it is 0% by definition. As can be seen, for all circuits flaw coverage is maximal when the power supply voltage is at 2.0 V. However, we already discovered that the non-nominal fault coverage is not optimal at 2.0 V. Therefore both maximal coverage of flaws and hard defects may not be realized at the same time.

In summary, if only one test application run can be afforded, low-voltage testing is efficient, yet power supply voltage should be lowered moderately, i.e., to 2.5 V, 2.8 V, or 3.0 V, rather than to the lowest considered value of 2.0 V.

Scenario AS

In Scenario **AS** we allow for one test application under the nominal conditions supplemented by an additional test run under the non-nominal conditions. In the latter run, low-temperature testing and low-voltage testing may be used in combination. We disregard the performance degradation implied by low-voltage testing, i.e. the complete test set is applied even at reduced power supply voltage.

In the experiments we considered two values for the nominal temperature T^{nom} : 370 K and 300 K. During testing, the device dissipates power which leads to increased junction temperature, unless the temperature is controlled during (nominal) testing using a thermal chuck. Nominal temperature of 300 K is valid when the temperature is controlled, whereas T^{nom} of 370 K holds if the temperature is not controlled. Note that the operating temperature of a packaged IC is in general closer to 370 K than to 300 K. For this experiment the nominal power supply voltage V_{dd}^{nom} is again 3.3 V. For the low-X test application, we considered voltages of 3.0 V, 2.8 V, 2.5 V, and 2.0 V. Furthermore, operating temperatures T of 300 K and 196 K (which is the evaporating temperature of nitrogen) were evaluated. For the experiments reported in [189], a nominal temperature of 373 K (100 °C) was taken, while 273 K (0 °C) was regarded as low temperature. By contrast we chose a higher value of 300 K for our “reduced-temperature” scenario, as it is less likely to lead to condensation problems.

When factoring the impact of low-temperature testing into the electrical part of the resistive bridging fault model, we considered the effects discussed in Chapter 9.1.1. All necessary parameters were taken from the SPICE model card provided for the AMS 0.35 μm technology. We assumed the defect material to be aluminum, which – for $T^{\text{nom}} = 370$ K and $T = 300$ K – resulted in a resistance reduction by a factor of 1.29 between nominal and low temperature.

Experimental results for a combination of nominal and low-X testing with $T^{\text{nom}} = 370$ K and $T = 300$ K can be found in Table 9.7. It features the name of the circuit in column one. Column two gives the fault efficacy $G\text{-FE}$ from Equation (5.4.7) for nominal temperature and voltage. In column three, we give $\text{FC}_{\text{comb}}^{\text{mn}}$ from Equation (9.1.2) which we obtained when lowering only the operating temperature. In addition to that, power supply voltage was lowered as well to derive $\text{FC}_{\text{comb}}^{\text{mn}}$ from columns 4–7. The last row of the table gives average coverages.

Table 9.1: Average $FC_{\text{comb}}^{\text{nn}}$ for different values of V_{dd} , T^{nom} , and T .

T^{nom} [K]	T [K]	$FC_{\text{comb}}^{\text{nn}}$				
		3.3 V	3.0 V	2.8 V	2.5 V	2.0 V
300	196	97.35	97.41	97.43	97.44	97.52
370	196	97.46	97.49	97.51	97.52	97.59
370	300	97.21	97.35	97.41	97.49	97.56
370	370	96.83	97.11	97.23	97.40	97.55

From Table 9.7 we can see that the coverage of hard defects is indeed increased by a combination of low-voltage and low-temperature testing. Yet, on average this increase is rather low and does not exceed 1%. We repeated the same experiment with several different temperature configurations – average values of $FC_{\text{comb}}^{\text{nn}}$ can be found in Table 9.1. The last row of the tables refers to testing at reduced power supply voltage but nominal temperature. Results confirm the observations made for $T^{\text{nom}} = 370$ K and $T = 300$ K. In summary, even though the combination of low-voltage and low-temperature testing increases the coverage of hard defects, this coverage gain is limited. In particular there is no advantage over testing at reduced supply voltage only. Given the fact that low-temperature testing is very cost-intensive, it does not seem to be a justifiable method to detect hard defects.

Contrary to that, a very high flaw coverage is obtained when combining low-voltage and low-temperature testing. Tables 9.8, 9.9, 9.10, and 9.11 list experimental data for $FC_{\text{flaw}}^{\text{nn}}$ and different configurations of T^{nom} and T . The first column of each table gives the name of the circuit followed by the flaw coverages derived for the considered power supply voltages in columns 2–6. The last row of the tables gives the respective average coverages. Additionally, averages are visualized in Figure 9.9. We may observe that low-X testing is very effective in detecting flaws. Most notably, the combination of low-voltage and low-temperature testing yields superior results of up to 92% flaw coverage. In particular, we may benefit from low-temperature testing if power supply voltage cannot be overly reduced. For instance, lowering the voltage from 3.3 V to 2.5 V while keeping the temperature constantly at 370 K yields approximately 60% of the flaws. However, almost the same flaw coverage can already be achieved for a V_{dd} of 3.0 V, if additionally the temperature is lowered to 196 K.

Discussion

Low-temperature and low-voltage testing differ substantially not only in their impact on the process of testing but also in terms of their influence on the defect detection. Low-voltage testing is very cost-effective as it increases the coverage of a given test set without affecting test cost. Neither additional test equipment nor dedicated design for testability logic is required for this method. Yet, testing at reduced power supply voltage may only be performed when operating frequency is lowered as well, and, thus, this technique prolongs test time. In contrast to that, low-temperature testing requires additional equipment to

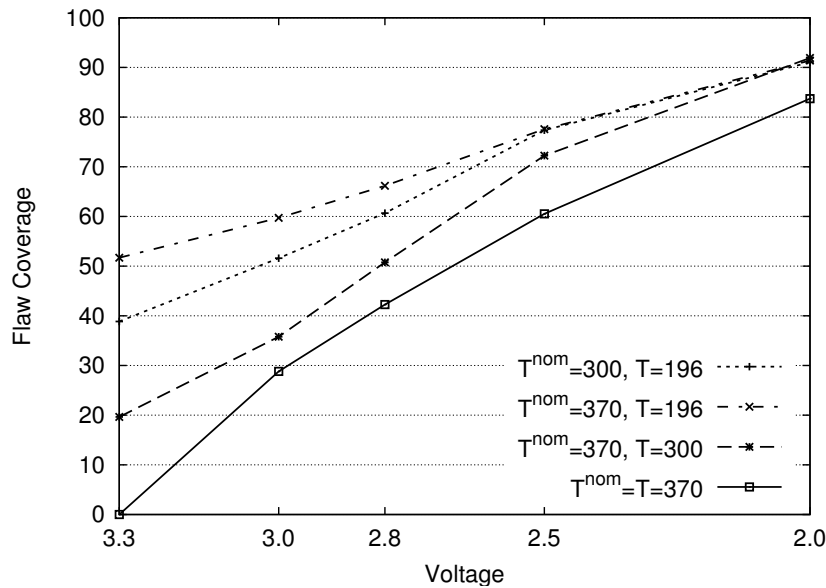


Figure 9.9: Average flaw coverage $FC_{\text{flaw}}^{\text{mn}}$ from Tables 9.8 – 9.11.

control the temperature during testing; this in turn implies additional cost. Test time, however, is not affected by this method.

Low-voltage testing is particularly well suited for the detection of hard defects. Our analysis demonstrated that this is especially true for larger test sets. Furthermore, we found that the recommendable value for V_{dd} decreases with increasing size of the test set. Low-temperature testing, on the other hand, only achieves mediocre coverage of hard defects. Even if combined with low-voltage testing, the high costs involved with temperature control may not be justified by its efficiency. Consequently, low-voltage testing alone is the method of choice to detect hard defects.

Equally, the coverage of flaws is very high when using low-voltage testing. Optimal results may be obtained if the power supply voltage is reduced to the lowest reasonable level, regardless of whether the performance degradation is accounted for. The coverage of flaws is also significantly improved by low-temperature testing. This suggests that combined low-voltage and low-temperature testing is the supreme method to detect dynamic defects and reliability defects by static (scan) test.

If test cost is extremely sensitive, low-temperature testing may not be acceptable. Hence, we are restricted to the application of a truncated test set under reduced power supply voltage only. Yet, voltage should only be moderately reduced to assure both optimal detection of hard defects and coverage of a large share of flaws. Extreme reduction of V_{dd} seems to be advisable only when an additional test run under nominal voltage and temperature can be expended. Similarly, low-temperature testing is most efficient when combined with a test run under the nominal conditions. This assures a largely improved coverage of hard defects and optimal flaw detection.

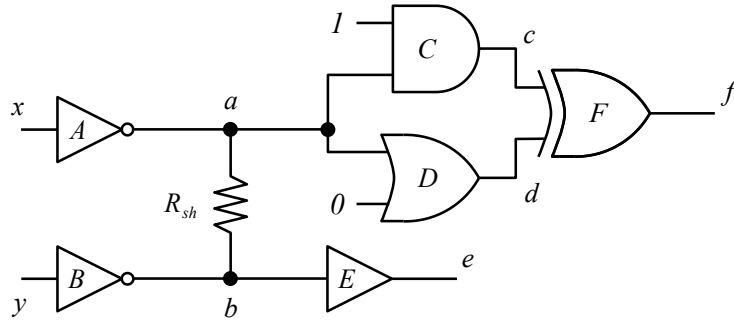


Figure 9.10: Example circuit with a resistive short affecting nodes a and b .

9.1.3 Coverage Loss by Low-Voltage Testing

In the research published so far it has always been argued that lowering the power supply voltage will extend the range of short resistances detected under the nominal conditions. We found, however, that in some situations this is not true and some defects detected at nominal V_{dd} may become undetectable when lowering the voltage.

Consider the circuit from Figure 9.10 and assume that the only valid logical assignment to inputs x and y , which activates the bridging fault, is $x = 0$ and $y = 1$. As both driving gates are inverters this results in $a = 1$ and $b = 0$. In a fault-free circuit these values translate into $c = d = 1$ (the side inputs of both driven gates C and D are fixed at non-controlling values) and consequently $f = 0$; furthermore $e = 0$ holds.

The voltage characteristics present at the shorted nodes in the faulty circuit are illustrated in Figure 9.11. Under the nominal power supply voltage, characteristics at a and b are V_a^{nom} and V_b^{nom} , respectively. We will focus on the characteristic at node a since V_b^{nom} (depicted in gray) does not have any intersection with the logic threshold $V_{\text{lt}}^{E,\text{nom}}$ of driven gate E and, thus, does not allow for detection of the defect. Logic thresholds of driven gates C and D are $V_{\text{lt}}^{C,\text{nom}}$ and $V_{\text{lt}}^{D,\text{nom}}$, respectively. We obtain two critical resistances R_C^{nom} and R_D^{nom} , with $R_C^{\text{nom}} < R_D^{\text{nom}}$. For a short defect with resistance $R_{\text{sh}} < R_C^{\text{nom}}$ both driven gates interpret the logical value 0, resulting in $c = d = 0$ and consequently $f = 0$. As this equals the fault-free logical value, a defect with a resistance from this range is not detected. For a short with resistance $R_C^{\text{nom}} < R_{\text{sh}} < R_D^{\text{nom}}$ it holds that $c = 1$, $d = 0$, which translates into $f = 1$. Therefore, a defect from this resistance range is detected. For $R_{\text{sh}} > R_D^{\text{nom}}$ all nodes in the circuit assume the values present in the fault-free circuit.

If we now reduce V_{dd} below its nominal value, the characteristics at nodes a and b shift as well as the logic thresholds of gates C , D , and E . The respective values are indicated by superscript “nn” instead of “nom”. Again, no faulty values are interpreted by gate E . Yet, we obtain two new critical resistances R_C^{nn} and R_D^{nn} at gates C and D , respectively. An analysis similar to the one performed for the nominal V_{dd} yields that the only short defects detectable under the non-nominal conditions have a resistance $R_C^{\text{nn}} < R_{\text{sh}} < R_D^{\text{nn}}$. Since $R_D^{\text{nom}} < R_C^{\text{nn}}$, the detection ranges under the nominal and the non-nominal conditions are disjoint. More precisely, this means that the short defects detected under nominal V_{dd}

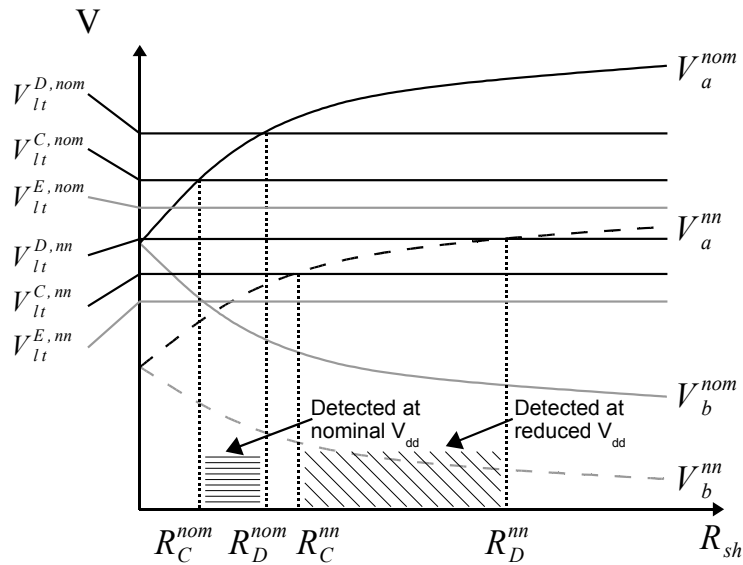


Figure 9.11: Voltage characteristics at nodes a and b as a function of R_{sh} for nominal (“nom”) and non-nominal (“nn”) operating conditions.

are no longer detectable when the power supply voltage is reduced, even though the very same test vector is applied in both cases. Nevertheless, the range of detected resistances has actually been enlarged by lowering V_{dd} and previously undetectable defects are now detected. Due to our assumptions, no other test vectors are able to cover the resistance range $[R_C^{nom}, R_D^{nom}]$, hence, defects from this range are redundant under the non-nominal conditions.

Coverage loss due to low-voltage testing has been observed in the mathematical analysis of Renovell et al. [156] as well. They found that this phenomenon may only occur if transistors with very specific width/length parameters are utilized in the driving gates. Yet, these kinds of transistors are – according to [156] – not typical for standard circuit design. The mechanism described here is completely different. In fact, lowering V_{dd} has indeed increased the range of resistances in which the driven gates interpret faulty logical values, just as conventional wisdom claims. However, it is the propagation path that is responsible for the coverage loss. Consequently, circuits designed according to standard principles are not immune to this type of coverage loss caused by low-voltage testing.

A similar instance of this problem has been observed by Sar-Dessai et al. [165]. They report on an electrical simulation experiment in which lowering V_{dd} resulted in coverage loss which could be traced back to a specific circuit structure: A fault site where there exists only a single driven gate which is fed by both driving gates. Yet, according to [165] this situation is very rare.

Metric for Coverage Loss and Experimental Results

In order to quantify the coverage loss due to the abovementioned phenomenon, we introduce the *coverage loss* metric. It focuses on those defects which are detectable under the nominal

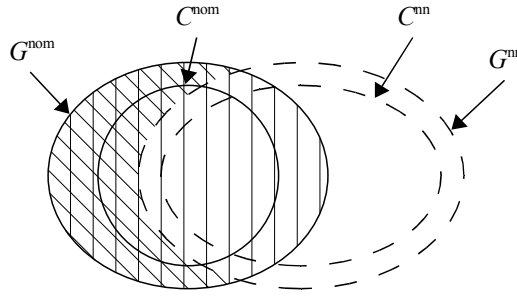


Figure 9.12: Venn diagram for coverage loss $FC_{\text{loss}}^{\text{nn}}$ (Equation (9.1.5)). Diagonal lines indicate the numerator, vertical lines show the denominator.

Table 9.2: Coverage loss $FC_{\text{loss}}^{\text{nn}}$ (circuits with $FC_{\text{loss}}^{\text{nn}} = 0\%$ for all V_{dd} settings excluded).

Circuit	3.0 V		2.8 V		2.5 V		2.0 V	
	#F	$FC_{\text{loss}}^{\text{nn}}$	#F	$FC_{\text{loss}}^{\text{nn}}$	#F	$FC_{\text{loss}}^{\text{nn}}$	#F	$FC_{\text{loss}}^{\text{nn}}$
c3540	1	0.0003	1	0.0004	1	0.0004	0	0.0000
cs00208	0	0.0000	0	0.0000	0	0.0000	2	0.0222
cs00420	0	0.0000	0	0.0000	0	0.0000	4	0.0176
cs00510	0	0.0000	0	0.0000	0	0.0000	32	0.1404
cs00838	0	0.0000	0	0.0000	0	0.0000	1	0.0044
cs09234	2	0.0125	2	0.0201	2	0.0201	2	0.0201
cs15850	1	0.0003	1	0.0003	1	0.0003	0	0.0000
cs38417	2	0.0009	2	0.0013	0	0.0000	0	0.0000

conditions but cannot be detected by an optimal test set under the non-nominal conditions ($R_{\text{sh}} \in G^{\text{nom}} \setminus G^{\text{nn}}$). This range is related to the (non-empty) interval of resistances detectable under the nominal conditions. The metric is defined as follows (see Figure 9.12 for an illustration):

$$FC_{\text{loss}}^{\text{nn}} = 100\% \cdot \frac{\int_{(G^{\text{nom}} \setminus G^{\text{nn}})} \rho(r) dr}{\int_{G^{\text{nom}}} \rho(r) dr}. \quad (9.1.5)$$

Experimental results can be found in Table 9.2. We retained the same setup used for the experiments from Chapter 9.1.2. For all circuits in Table 9.2 10,000 faults were simulated, with the exception of cs00208, for which we considered 3,986 bridging faults. The name of the circuit is found in column one. Columns two to nine give the number of faults “#F” for which coverage loss was observed followed by the average $FC_{\text{loss}}^{\text{nn}}$ for a V_{dd} of 3.0 V, 2.8 V, 2.5 V, and 2.0 V. Note that we excluded all circuits for which no coverage loss was observed at any low-voltage setting. It may be seen that coverage loss indeed occurs. Yet, only a very small share (at most 32 out of 10,000) of these faults is affected by coverage loss. Furthermore, $FC_{\text{loss}}^{\text{nn}}$ is extremely small in magnitude.

9.2 Delta-IDDQ Testing

For many years it is feared that the ever increasing and highly variable leakage currents in state-of-the-art CMOS circuits might render quiescent current (IDDQ) testing unfeasible [83]. IDDQ testing is a methodology to detect resistive shorts and other types of defects by monitoring the supply current (see [144] for a detailed survey). It is particularly well suited for CMOS circuits which draw (in theory) only very little current in steady-state, i.e. when the nodes are not switching. Therefore, an elevated supply current is an indicator for the presence of a defect. To be detectable by IDDQ testing, a resistive short only has to be activated – no propagation of fault effects is necessary. This facilitates test pattern generation for IDDQ testing. On the other hand, the evaluation of the quiescent current is very time-consuming, such that typically only a few IDDQ measurements can be afforded.

Unfortunately, it is no longer true that the current drawn by a defect-free CMOS circuit is negligible ($IDDQ \approx 0$) when the nodes are not switching. Rather, in recent circuits there is a significant *background IDDQ current* I_{back} flowing through the circuit regardless of whether it is defective. This current is composed of the (sub-threshold) leakage current I_{off} of the inactive transistors. Although the contribution of every individual transistor is minute, the cumulative effect of millions of transistors is considerable. Low- V_T transistors used in high-performance logic are particularly prominent contributors to I_{back} . Furthermore, due to e.g. process variations I_{back} may differ substantially from die to die. As a consequence, the conventional assumption of a single pass/fail threshold, which is universally valid for all dies, is no longer sufficient to discriminate between good and defective devices [203]. Nevertheless, IDDQ testing may still be used thanks to a combination of design measures and innovative test methodologies, such as e.g. current signatures [52], current ratios [111], and Delta-IDDQ [48, 119, 184, 185].

Delta- or differential IDDQ testing evaluates the difference in the amount of current drawn by a defective circuit depending on whether the defect is activated or not. Consider a short between two nodes a and b and a test set T containing two test vectors t_1 and t_2 . Assume that t_1 induces the same logical value at both nodes a and b (e.g. $a = b = 0$), and that t_2 induces opposite logical values at a and b (e.g. $a = 1$ and $b = 0$). Note that IDDQ testing only requires fault activation and that no fault effect propagation is necessary for detection. In Delta-IDDQ testing we first apply t_1 and measure quiescent current $I_{DDQ} = I_{back}$ for the inactive short. Then we apply the second vector t_2 , which activates the short and, thus, creates an unwanted connection between the pull-up network driving node a and the pull-down network driving node b . Through this connection additional current I_{defect} is flowing. Therefore, the quiescent current measured for t_2 is higher than that for t_1 . More precisely, I_{DDQ} consists of two components, the background current and the defect induced current, i.e. $I_{DDQ} = I_{back} + I_{defect}$. The relative increase in current observable when comparing vectors t_1 and t_2 is evaluated by Delta-IDDQ testing. If for different test vectors the difference in the quiescent current exceeds a predefined threshold ΔI_{limit} the circuit is said to be defective. If, however, differences in I_{DDQ} remain below ΔI_{limit} Ampere for all vectors applied, the circuit is said to defect-free.

There are essentially two criteria for the applicability of Delta-IDDQ testing to today’s advanced technologies. First of all, the current measurement equipment should be sufficiently fast to meet test time and cost constraints. Additionally, the equipment has to be sensitive enough to allow discrimination of I_{back} and $I_{\text{back}} + I_{\text{defect}}$. Secondly, it has to be assured that the test set used for Delta-IDDQ testing contains for each short under consideration one vector which activates the defect and one vector which does not activate the defect [139]. All shorts for which this requirement is not met, i.e. which are either always or never activated, are not detected by Delta-IDDQ testing. An additional, yet optional “requirement” is that the variation of I_{DDQ} is small compared to I_{back} , as this affects the screening efficiency. This might not be the case for high-performance parts such as the largest microprocessors on the market. Yet, for many other products, including low-power designs which contain high- V_T transistors with well-controlled leakage, Delta-IDDQ testing is still applicable [83, 94]. This is also true, if I_{back} is subject to intra-die variations, as the value of ΔI_{limit} is rather die related and not necessarily constant for a whole production lot. For the same die, background current may also vary depending on the test vector applied. This is as the number of inactive transistors differs from test vector to test vector [113]. Fortunately, the average number of inactive transistors tends to stabilize with increasing number of transistors. Consequently, the deviation between minimum and maximum I_{back} of a defect-free device is rather small for large ICs [111]. This simplifies the choice of ΔI_{limit} and ensures the applicability of Delta-IDDQ testing.

In the following we will investigate the detection of resistive shorts using Delta-IDDQ testing. To the best of our knowledge, this is the first study of its kind which is performed within the framework of the resistive bridging fault model. The contents of this chapter have been adapted from our publications [W10] and [P13]. We will focus on the detection of *active defects* [52], i.e. defects which affect switching nodes (as opposed to passive nodes such as V_{dd} and ground terminals) of the circuit, only. Furthermore, we will, as usual, consider solely resistive inter-gate shorts (IDDQ testing of intra-gate shorts is covered e.g. in [160, 190]). We will extend the resistive bridging fault model to be able to compute short resistance intervals detected by Delta-IDDQ testing. Several metrics are introduced which help to exactly quantify the effectiveness of resistive short detection by Delta-IDDQ testing, voltage testing and the combination of both techniques. The metrics thoroughly discriminate between shorts detectable by voltage testing (hard defects) and those detected by Delta-IDDQ testing only (flaws).

Some authors claim that when applying IDDQ testing to sequential circuits it is safe to use the fault-free next state function to determine the activation of a fault [183]. Yet, as was noted by [99, 134, 160] this may lead to incorrect results for certain shorts which modify the values stored in flip-flops. We demonstrate that this effect also applies to resistive bridging faults and that our approach is able to correctly resolve these non-trivial situations. Beyond that, our study shows that there are circuit configurations in which the interval of short resistances detected by Delta-IDDQ testing is non-contiguous. This is contrary to conventional wisdom, which suggests, that the detection ranges may only be contiguous. Our results underline that for sequential circuits this conventional assumption is not always true, and that fault effects stored in flip-flops may prevent the detection of certain shorts by Delta-IDDQ testing.

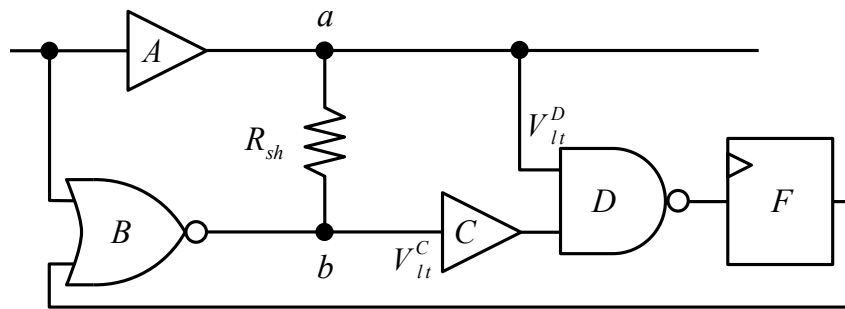


Figure 9.13: Example of a sequential circuit with a resistive short affecting nodes a and b .

Our experimental data indicates that Delta-IDDQ testing is very valuable to detect resistive shorts detectable by voltage testing and, in addition to that, adds coverage of flaws, undetectable by conventional testing. This also applies to test sets which are considerably smaller than those used for voltage testing. However, as we demonstrate, test sets have to conform with the requirements of Delta-IDDQ testing to guarantee superior results. Our experiments, give indications on the improvements in coverage that may be obtained when respecting these requirements. Furthermore, we show that a combination of Delta-IDDQ and voltage testing may increase the coverage of defects beyond what may be detected by either method individually.

In the subsequent Chapter 9.2.1, we will introduce the extensions of the resistive bridging fault model required to integrate Delta-IDDQ testing and to quantify its effectiveness. Then, Chapter 9.2.2 will discuss the particularities of resistive shorts in sequential circuits. Experimental results are reported in Chapter 9.2.3. Finally, conclusions can be found in Chapter 9.3.

9.2.1 Extensions of the Resistive Bridging Fault Model

This chapter will first demonstrate the application of Delta-IDDQ testing to a small example circuit. In particular, we will transfer the concepts of critical resistance and detectability interval to this testing methodology. Subsequently, we will rephrase the general framework of our electrical model to enable the computation of critical resistances for Delta-IDDQ testing. The effectiveness of quiescent current testing may be quantified and related to the coverage of conventional voltage testing using the metrics introduced in the last part of this chapter.

Analogue Detectability Intervals for Quiescent Current Testing

Consider the circuit from Figure 9.13. It depicts a short between the output a of buffer A and the output b of NOR gate B . The shorted nodes are observed by driven gates C and D and, possibly, some other gate driven by node a not shown in the figure. Gate F represents some arbitrary flip-flop. Assume that the flip-flop stores the logical value 0 and

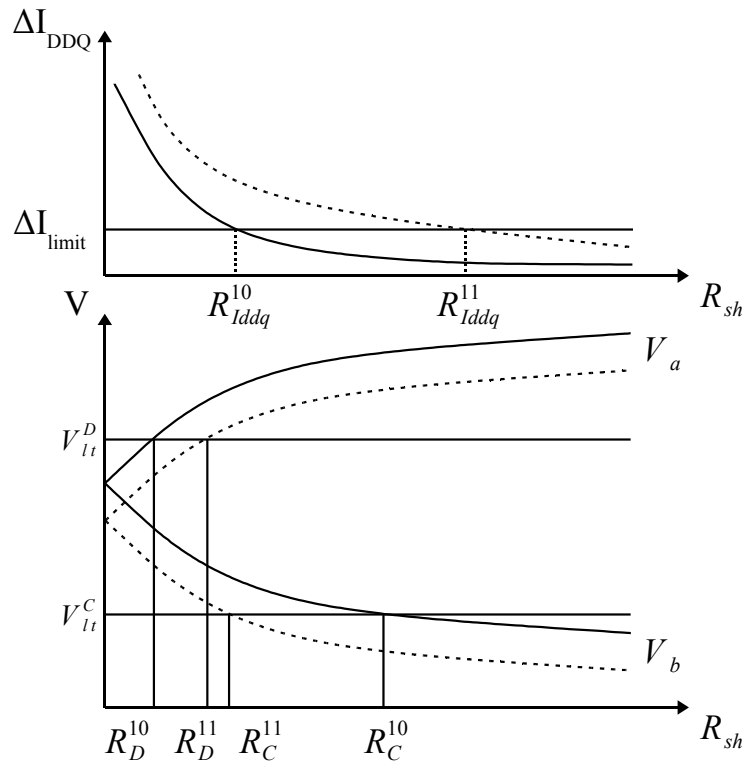


Figure 9.14: Current flowing through the defect (upper diagram) and voltage characteristic at nodes a and b (lower diagram) as a function of R_{sh} for two test vectors.

the logical value 1 is applied to the only input of the circuit. Obviously, in this case, $a = 1$ and $b = 0$ will hold. In a fault-free circuit this will result in the logical value 1 being driven at the output of gate D .

The voltage characteristics at nodes a and b in the presence of the short of resistance R_{sh} are depicted as a solid line in the lower diagram of Figure 9.14. Logic thresholds are V_{lt}^C of gate C and V_{lt}^D of gate D . The characteristic cross the logic thresholds of the respective driven gates and the projection of these intersections to the x -axis yields two critical resistances, R_C^{10} and R_D^{10} . (Note that superscript “10” indicates that these critical resistances are obtained when assigning the pattern (1, 0) to the inputs of gate B .) Consequently, the faulty logical value 1 will be interpreted by gate C for $R_{sh} < R_C^{10}$ and the faulty logical value 0 will be interpreted by gate D for $R_{sh} < R_D^{10}$. Due to the logic function of gate D , the gate’s output will be at the faulty logical value 0 in presence of a short with resistance $R_D^{10} < R_{sh} < R_C^{10}$ and at the fault-free logical value 1, otherwise.

The solid curve in the upper diagram of Figure 9.14 depicts the same situation from the perspective of current. It shows the characteristic of a defect induced current flowing through the short as a function of R_{sh} . We indicate the Delta-IDDQ threshold ΔI_{limit} as a horizontal line. Similar to the case of voltage testing, we may observe an intersection of the current characteristic with ΔI_{limit} . The projection of this intersection to the abscissa yields the critical resistance R_{iddq}^{10} . For any short having a resistance lower than R_{iddq}^{10} , i.e. $R_{sh} < R_{iddq}^{10}$, the current flowing through the defect exceeds ΔI_{limit} . Consequently, we

may detect the defect by Delta-IDDQ testing. For any short having a resistance larger than R_{iddq}^{10} , i.e. $R_{\text{sh}} > R_{\text{iddq}}^{10}$, the defect induced current is smaller than ΔI_{limit} and the short will remain undetected by Delta-IDDQ testing. Since in quiescent current testing no propagation is required, the interval in which ΔI_{limit} is exceeded corresponds to the detection interval. Similar to the ADI for voltage testing of resistive bridging faults we will denote the interval of short resistances detected by (Delta-)IDDQ testing as *ADI for IDDQ testing I-ADI*. In our example, the *I-ADI* obtained with Delta-IDDQ testing is $[0, R_{\text{iddq}}^{10}]$.

Now assume, that the flip-flop stores the logical value 1 instead of the logical value 0 and we continue to apply the logical value 1 to the input of the circuit. In this case two n -transistors are active in the pull-down network of gate B compared to only one in the situation discussed above. The driving strength of gate A remains unchanged and, thus, the voltage characteristics at nodes a and b may shift down. They are depicted as a dashed line in the lower diagram of Figure 9.14. As a consequence, we may record two new critical resistances R_C^{11} and R_D^{11} for gates C and D , respectively (superscript “11” denotes that now the pattern (1, 1) is assigned to the inputs of gate B). In this case, the output of gate D will see the faulty logical value 0 for an R_{sh} from the interval $[R_D^{11}, R_C^{11}]$ and the fault-free logical value 1, otherwise.

Similar to the case of voltage testing, an increase in the driving strength of a gate B influences the current flowing through the defect site. In our example we may observe an increased current depicted as a dashed line in the upper diagram of Figure 9.14. Analogous to voltage testing we derive the new critical resistance R_{iddq}^{11} for Delta-IDDQ testing. Hence, we obtain *I-ADI* $[0, R_{\text{iddq}}^{11}]$.

Electrical Model

In this chapter we will introduce a methodology to compute critical resistances for Delta-IDDQ testing. In quiescent current testing our interest is focused on the increased current flow which is caused by the unintended conducting path created by an activated short. Due to this path, a faulty instance of a circuit draws more current than a fault-free instance of the same circuit. This relative current increase can be measured and compared to a reference value ΔI_{limit} . Circuits for which ΔI_{limit} is exceeded are considered to be defective. Similarly to the voltage testing case, the critical resistance for Delta-IDDQ testing is defined as the maximum short resistance for which the current flowing through the unintended current path created by that short exceeds ΔI_{limit} .

The increased current flow may be ascribed to two effects. On the one hand the activated defect itself creates a current path which allows additional current to be drawn. On the other hand the activated defect may cause additional current to flow through the driven gates (see [9] for a discussion). This is, because intermediate voltage levels may be present at the shorted nodes due to the defect (an effect which is exploited by voltage testing). As a consequence both pull-up and pull-down network may be active in some of the driven gates (according to [162] this effect may even affect gates succeeding the driven gates). This creates additional, unintended current paths, which contribute to the overall current

increase. We take a conservative position and do not account for the secondary current increase caused by the driven gates. Thus, our fault coverage figures may be considered a pessimistic lower bound.

To determine the critical resistance for Delta-IDDQ testing, the general framework introduced in Chapter 5.2 to calculate critical resistances for voltage testing has to be rephrased. In the system of equations used in this framework (refer again to Equation (5.2.1)) the current flowing through the defect is denoted as I_0 . This is illustrated in Figure 5.3 for the case of two inverters. The current I_0 contributes to the overall quiescent current drawn by the circuit. We assume I_{back} to be independent of the defect-induced current. Hence, it is regarded as a constant offset which does not influence defect detection. Consequently, our analysis is valid for Delta-IDDQ testing.

Given ΔI_{limit} we are looking for the maximal short resistance R_{sh} for which the current flowing through the defect I_0 exceeds this limit, i.e. $I_0 > \Delta I_{\text{limit}}$ holds. Since for increasing R_{sh} we can expect I_0 to monotonically decrease, setting $I_0 = \Delta I_{\text{limit}}$ yields this critical resistance. The short resistance is calculated – according to Equation (5.2.1) – as: $R_{\text{sh}} = (V_{n_1} - V_{n_0})/I_0$. Voltages V_{n_1} and V_{n_0} can be obtained for a given I_0 using the inverse output characteristics of p -transistors (I_p^{-1}) and n -transistors (I_n^{-1}), respectively. Thus, for $V_{n_1} = I_p^{-1}(I_0)$ and $V_{n_0} = I_n^{-1}(I_0)$ we obtain:

$$R_{\text{sh}} = \frac{I_p^{-1}(I_0) - I_n^{-1}(I_0)}{I_0}. \quad (9.2.1)$$

Now we can set $I_0 = \Delta I_{\text{limit}}$ to determine the critical resistance for Delta-IDDQ testing $R_{\text{crit}}^{\text{IDDQ}} = R_{\text{sh}}$. Using the inverse functions of the output characteristics for the Shockley model, Equation (9.2.1) transforms to:

$$R_{\text{crit}}^{\text{IDDQ}} = \frac{1}{\Delta I_{\text{limit}}} \left(|V_{tp0}| + \sqrt{(V_{\text{dd}} - |V_{tp0}|)^2 - \frac{2 \cdot \Delta I_{\text{limit}}}{\mu_p C_{\text{ox}} W_p / L_p}} \right. \\ \left. - V_{\text{dd}} + V_{tn0} + \sqrt{(V_{\text{dd}} - V_{tn0})^2 - \frac{2 \cdot \Delta I_{\text{limit}}}{\mu_n C_{\text{ox}} W_n / L_n}} \right). \quad (9.2.2)$$

Refer to Table 5.2 for an explanation of the symbols used.

Obviously, the same methodology may also be used to derive similar equations for the Fitted or the Predictive Model. We do not account for additional increased current flow through the driven gates. In this sense Equation (9.2.1) (and thus Equation (9.2.2) as well) derives a pessimistic lower bound for the critical resistance of Delta-IDDQ testing.

Fault Coverage Metrics

To be able to accurately compare voltage testing and Delta-IDDQ testing we define several new fault coverage metrics which are introduced in this chapter. When evaluating the detection of resistive shorts by either method, it is crucial to carefully distinguish between

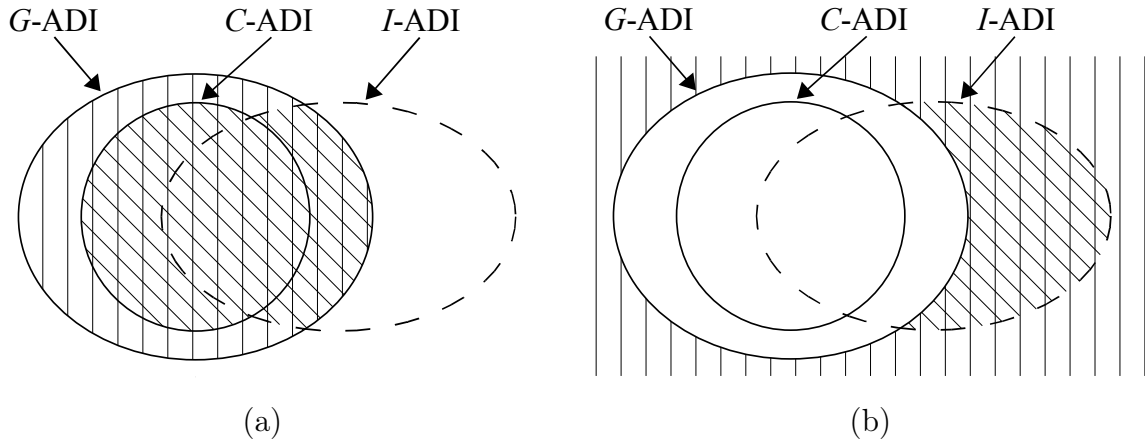


Figure 9.15: Venn diagram for (a) combined fault coverage $FC_{\text{comb}}^{\text{IDDQ}}$ (Equation (9.2.4)) and (b) flaw coverage $FC_{\text{flaw}}^{\text{IDDQ}}$ (Equation (9.2.5)). Diagonal lines indicate the numerator, vertical lines show the denominator.

defects detectable by voltage testing, i.e. $R_{\text{sh}} \in G\text{-ADI}$, and those only detected by Delta-IDDQ testing ($R_{\text{sh}} \in [0\Omega, \infty] \setminus G\text{-ADI}$). We will refer to the latter category of defects as flaws. As usual, all metrics are averaged if more than one fault is considered.

The first metric, which is defined for a single resistive bridging fault f , measures the coverage of shorts by quiescent current testing only. The detection of flaws is not accounted for, as we focus on short resistances contained in $G\text{-ADI}(f)$. Let $I\text{-ADI}(f)$ denote the union of all resistance ranges for which resistive bridging fault f is detected by the individual vectors of a given test set using quiescent current testing. The *IDDQ fault coverage* FC^{IDDQ} is defined as:

$$FC^{\text{IDDQ}}(f) = 100\% \cdot \frac{\int_{(I\text{-ADI}(f) \cap G\text{-ADI}(f))} \rho(r) dr}{\int_{G\text{-ADI}(f)} \rho(r) dr}. \quad (9.2.3)$$

To estimate the efficiency of the combination of voltage and quiescent current testing, we determine the range of short resistances detected by either technique, i.e. $R_{\text{sh}} \in C\text{-ADI}(f) \cup I\text{-ADI}(f)$. Note that again we are restricted to the range of defects detectable by voltage testing. The *combined fault coverage* metric $FC_{\text{comb}}^{\text{IDDQ}}$ is illustrated in the form of a Venn diagram in Figure 9.15(a); it computes as follows:

$$FC_{\text{comb}}^{\text{IDDQ}}(f) = 100\% \cdot \frac{\int_{((C\text{-ADI}(f) \cup I\text{-ADI}(f)) \cap G\text{-ADI}(f))} \rho(r) dr}{\int_{G\text{-ADI}(f)} \rho(r) dr}. \quad (9.2.4)$$

All metrics introduced so far restricted the detected short resistance range to those defects detectable by voltage testing. The detection of flaws, only detectable by quiescent current testing, is captured by the *IDDQ flaw coverage* metric $FC_{\text{flaw}}^{\text{IDDQ}}$. It computes the range of resistances detected by voltage or quiescent current testing which are undetectable by

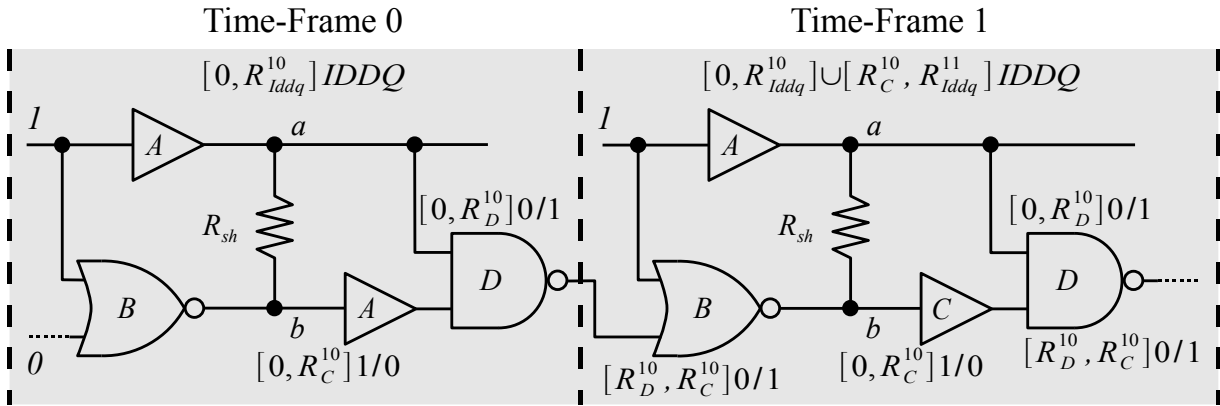


Figure 9.16: Two-frame time-frame expansion of the circuit from Figure 9.13 (I -ADIs are marked by “IDDQ”).

voltage testing, i.e. $([0 \Omega, \infty] \setminus G\text{-ADI}(f)) \cap I\text{-ADI}(f)$. This is related to the range of all flaws – see Figure 9.15(b) for an illustration. The definition of the metric is:

$$FC_{\text{flaw}}^{\text{IDDQ}}(f) = 100\% \cdot \frac{\int_{([0 \Omega, \infty] \setminus G\text{-ADI}(f)) \cap I\text{-ADI}(f)} \rho(r) dr}{\int_{[0 \Omega, \infty] \setminus G\text{-ADI}(f)} \rho(r) dr}. \quad (9.2.5)$$

Assume for example that $G\text{-ADI}$ is $[0, 1000]$ and that $I\text{-ADI}$ is $[0, 1500]$. Then the numerator of Equation (9.2.5) will be $[1000, 1500]$. The denominator, on the other hand, will be $[1000, \infty]$.

Note that $FC_{\text{flaw}}^{\text{IDDQ}}$, $FC_{\text{comb}}^{\text{IDDQ}}$, and $FC_{\text{flaw}}^{\text{IDDQ}}$ are closely related to metrics FC^{nn} , $FC_{\text{comb}}^{\text{nn}}$, and $FC_{\text{flaw}}^{\text{nn}}$, defined in Chapter 9.1.1.

9.2.2 Delta-IDDQ Testing of Sequential Circuits

Refer again to the circuit from Figure 9.13 and assume that the logical value 1 is applied in two consecutive time-frames to the input of the circuit. Assume furthermore that before the first application of a test, the flip-flop F is initialized to the logical value 0. Our preceding analysis from Chapter 9.2.1 yielded two critical resistances R_C^{10} and R_D^{10} for the case of voltage testing and critical resistance R_{iddq}^{10} when Delta-IDDQ testing is performed (refer to Figure 9.14). Similarly, if the logical value 1 is stored in the flip-flop our analysis demonstrated that the critical resistances R_C^{11} , R_D^{11} , and R_{iddq}^{11} may be obtained. We assume $0 < R_D^{10} < R_{\text{iddq}}^{10} < R_C^{10} < R_{\text{iddq}}^{11}$.

Figure 9.16 depicts the unrolled version of the circuit for the two consecutive time-frames, i.e. the circuit’s two-frame time-frame expansion (refer to Chapter 2.2.2 for an introduction of this technique). From the preceding discussion we know that in time-frame 0, i.e. when the logical value 0 is present in the flip-flop, due to a short with an intrinsic resistance from the range $[R_D^{10}, R_C^{10}]$, the logical value 0 will be stored instead of the fault-free logical

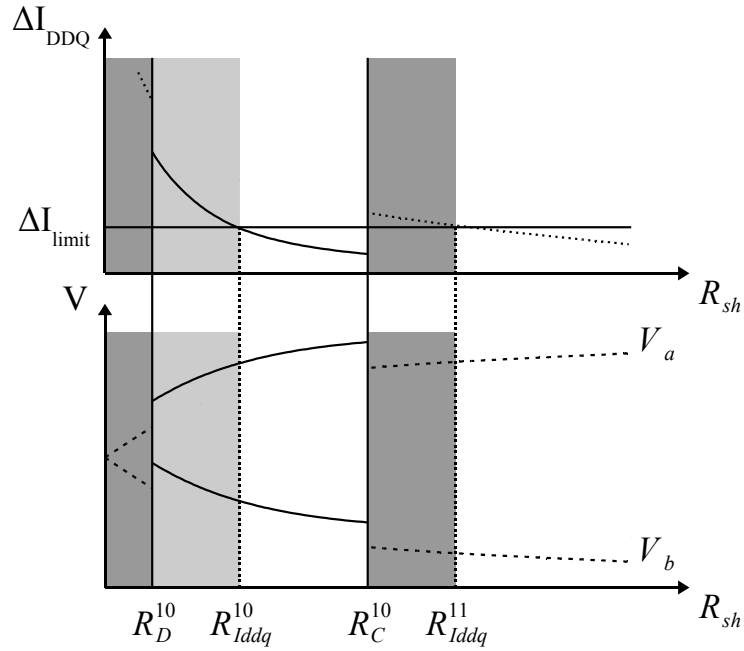


Figure 9.17: Detection conditions for the second time-frame. Resistance ranges highlighted in (light) gray are detected by Delta-IDDQ testing.

value 1. As a consequence, in time-frame 1 the fault-free logical value 1 will be fed into the circuit only if for the short's resistance either $R_{sh} < R_D^{10}$ or $R_C^{10} < R_{sh}$ holds.

In time-frame 1 the characteristics of voltage and current, respectively will match those illustrated by solid lines in Figure 9.14 only if $R_{sh} \in [R_D^{10}, R_C^{10}]$. For all other short defect resistances we have to refer to the characteristics illustrated by dashed lines in the figure; this is clarified in Figure 9.17. From our prior analysis it is clear that if the logical value 1 is stored in the flip-flop and we assign the logical value 1 to the input of the circuit, Delta-IDDQ testing allows the detection of defects from $[0, R_{Iddq}^{11}]$. Unfortunately, in time-frame 1 the complementary logical value 0 is supplied for $R_{sh} \in [R_D^{10}, R_C^{10}]$. Short defects with a resistance from this range may only be detected by Delta-IDDQ testing if their resistance is below R_{Iddq}^{10} . Thus, in summary, Delta-IDDQ detects shorts only if their resistance is within $[0, R_{Iddq}^{10}] \cup [R_C^{10}, R_{Iddq}^{11}]$. This is illustrated in Figure 9.17 by the resistance ranges highlighted in gray (where light gray marks those ranges detected when the logical value 0 is stored in the flip-flop F). Obviously this range is neither contiguous nor does it correspond to the interval $[0, R_{Iddq}^{11}]$ obtained when assuming the fault-free logical value 1 in F in time-frame 1 (as suggested in [183]). Adopting this simplified assumption would mean that defects with $R_{sh} \in [R_{Iddq}^{10}, R_C^{10}]$ were spuriously declared as detected by Delta-IDDQ.

As was noted in [159], the behavior described above is unique for shorts with non-zero resistance. For resistive shorts, the analysis framework proposed in this chapter is essential to identify the exact conditions under which the standard assumption of a fault-free next state is invalid. It would be impossible to derive this behavior without considering the short resistance explicitly.

9.2.3 Experimental Results

We applied 1,000 random test vectors to the ISCAS 85 [19] and the combinational cores of the ISCAS 89 [18] benchmark circuits. The fault list contained 10,000 randomly selected non-feedback bridging faults, where available.⁸ Alternatively, we could also use layout extracted bridging faults (refer to Chapter 3.3). We employed the Shockley technology model and parameters from the SPICE model card of a 0.35 μm technology from austriamicrosystems AG to determine the critical resistances (for voltage testing the Shockley Model equations from Chapter 5.2.2 were employed). When calculating fault coverages we used the distribution ρ proposed in [165] which is based on data from [158]. Yet, all experiments in this chapter could be easily repeated with any other short resistance distribution. The Delta-IDDQ threshold ΔI_{limit} was set to 100 μA . This is a typical resolution of high-current ($> 100 \mu\text{A}$) IDDQ measurement systems although better-resolving systems are available. We did not observe much variability in results using ΔI_{limit} of 50 μA and even 10 μA . All experiments were performed with an extended version of the simulator from [J3] which ran on a 2 GHz Pentium IV with 2 GB RAM.

The main objective of our experiments was to provide insight into the detection capabilities of Delta-IDDQ testing only, and those of a combination of Delta-IDDQ and voltage testing. In general, the number of (Delta-)IDDQ measurements is substantially lower than the number of test vectors applied when using voltage testing. Hence, (Delta-)IDDQ measurements are only performed for a subset of the vectors used for voltage testing. To account for this fact, we conducted (Delta-)IDDQ experiments for 10, 100, and 1,000 out of 1,000 random vectors contained in the test set for voltage testing.

Delta-IDDQ testing may only be performed if the background current I_{back} of the device under test is known. In order to obtain this reference value, there has to be at least one test vector for which the defect does not contribute to the overall quiescent current. This implies that for each resistive bridging fault tested there must at least one vector in the test set which does not activate the fault. If for any fault this requirement is not met, i.e. all vectors impose opposite values on the nodes shorted by the fault, this fault remains undetected by Delta-IDDQ testing. In the following we will refer to faults which are activated for all test patterns in the test set as *always activated* (AA) faults. Typically, the number of always activated faults tends to be higher for smaller test sets, such as the one containing only 10 test vectors. We evaluated the magnitude of AA faults by calculating the number of bridging faults which were activated under all IDDQ measurements. Furthermore, all AA faults have been excluded when calculating fault coverages for Delta-IDDQ testing. By contrast, no restrictions concerning AA faults apply to voltage testing. Thus, all resistive shorts detected by voltage testing were considered when calculating fault coverages.

The experimental results obtained for 10, 100, and 1,000 IDDQ measurements can be found in Tables 9.12, 9.13, and 9.14, respectively. Note that the test sets used for voltage testing always contained 1,000 random vectors. All tables have the same outline. In column one the name of the circuit is given. Column two states global fault efficacy $G\text{-FE}$ from

⁸Note that unless otherwise specified the fault lists used in this chapter are equal to those employed in all other experiments discussed in Part II of this work.

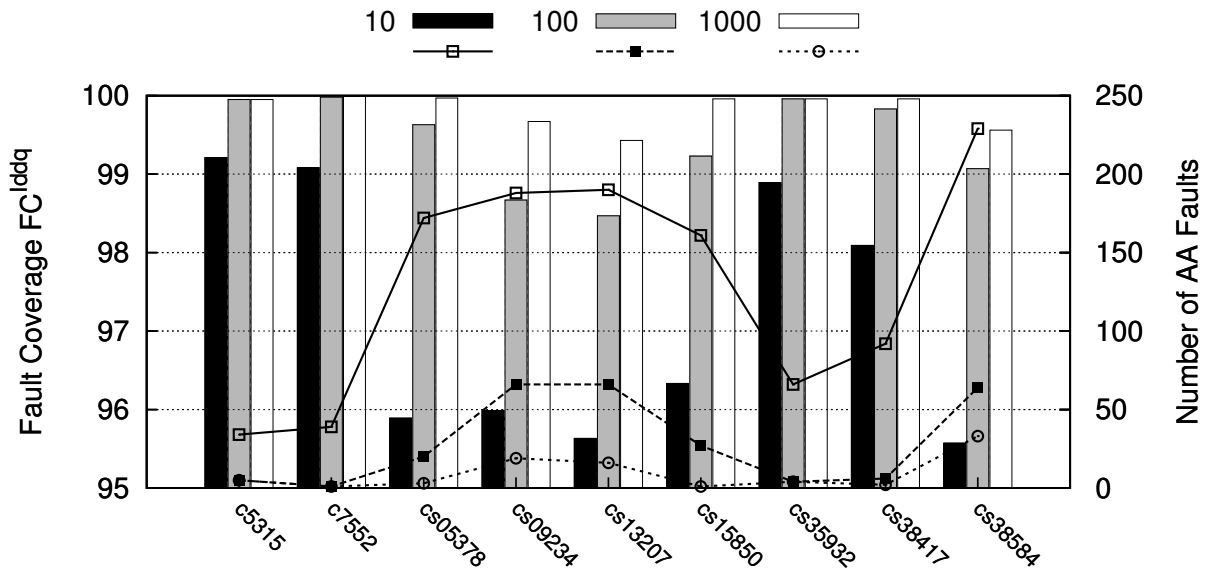


Figure 9.18: IDDQ fault coverage FC^{IDDQ} (bar graph) and number of AA faults (line graph) for 10, 100, and 1,000 Delta-IDDQ measurements.

Equation (5.4.7) obtained for logic testing only. The following four columns contain results of the Delta-IDDQ measurements. The number “AA” of always activated faults is given in column three. Fault coverage FC^{IDDQ} computed for Delta-IDDQ testing only (with AA faults excluded) can be found in column four. The combination of both techniques is evaluated by the combined fault coverage FC_{comb}^{IDDQ} which is given in column five. Subsequently, column six contains the IDDQ flaw coverage FC_{flaw}^{IDDQ} . Recall that by definition, the flaw coverage of voltage testing is 0%.

As may be seen, FC^{IDDQ} obtained by solely performing Delta-IDDQ measurements is strongly correlated to the number of always activated faults. A large number of AA faults directly implies low FC^{IDDQ} and vice versa. This is illustrated by Figure 9.18 for a selection of large circuits. The chart depicts FC^{IDDQ} (as a bar graph) and the number of AA faults (as a line graph) for 10, 100, and 1,000 Delta-IDDQ measurements. The effectiveness of Delta-IDDQ testing improved significantly when combined with voltage testing – in particular, if only a few IDDQ measurements were allowed. To a certain extent, this may be attributed to the fact that voltage testing detected some of the always activated faults which were missed by Delta-IDDQ testing. Yet, even for 1,000 measurements the additional voltage testing run still improved the average coverage of detectable defects from 99.42% to 99.74%. Voltage testing alone, however, achieved an average fault coverage of only 97.32%. This suggests that Delta-IDDQ testing detects defects not detected by voltage testing and vice versa. We also observed extremely high flaw coverages, which are again closely correlated to the number of AA faults.

To further investigate the impact of AA faults, we recomputed FC^{IDDQ} and FC_{comb}^{IDDQ} assuming that both AA and non-AA faults are detectable. Results can be found in the last two

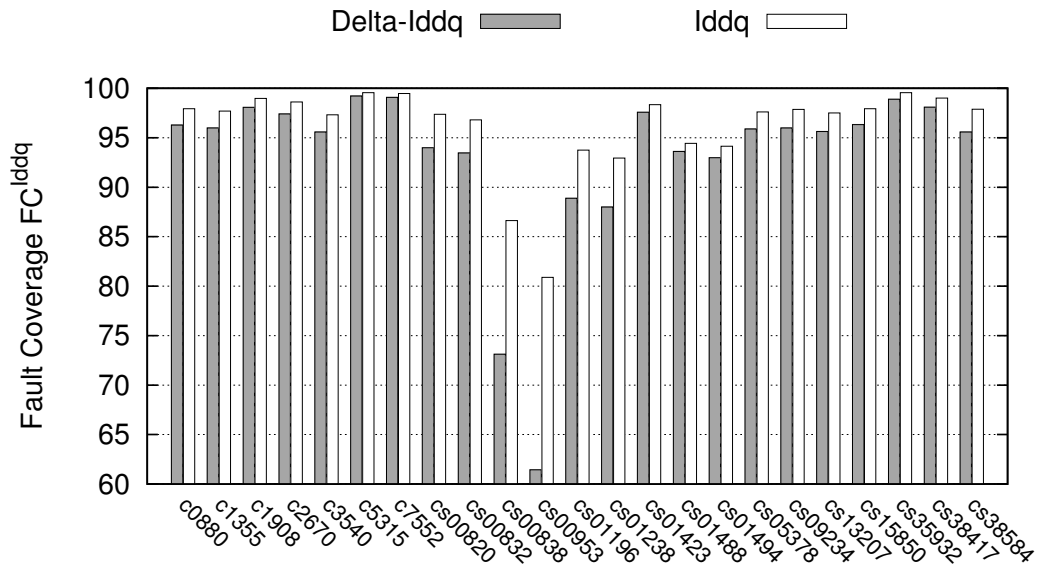


Figure 9.19: IDDQ fault coverage FC^{IDDQ} for Delta-IDDQ and IDDQ testing (10 IDDQ measurements).

columns of Tables 9.12, 9.13, and 9.14 which are labeled “IDDQ” (FC^{IDDQ} in column seven and $FC_{\text{comb}}^{\text{IDDQ}}$ in column eight). Note that these figures are mainly given for reference and their physical meaning is limited. The numbers would have been correct if IDDQ (not Delta-IDDQ) testing had been employed. However, in contrast to Delta-IDDQ testing, IDDQ testing with a limit of only $100 \mu\text{A}$ does not appear to be realistic. To verify our results, we also ran experiments assuming a much higher threshold of up to $1,500 \mu\text{A}$, which would be appropriate for regular (not Delta-) IDDQ testing. We found that this meant a drastic reduction in fault coverage.

The additional results in the tables show that the coverage deterioration, which may be accounted to the AA faults, is significant. This effect is most severe if only a few IDDQ measurements are allowed: For 10 measurements fault coverage FC^{IDDQ} leaps from an average of 93.48% to 96.48% if AA faults are not excluded (individual results are illustrated in Figure 9.19). This implies that AA faults must be accounted for when establishing the quality of a test set. Furthermore, the results show that a sufficient number of IDDQ tests is required to achieve a high Delta-IDDQ fault coverage. Also, the need for special test sets optimized to avoid AA faults and for ATPG tools to produce such test sets is obvious. In this sense, the fault coverages obtained without elimination of AA fault detections may be seen as an indicator for the efficiency to be expected from these improved test sets.

9.3 Conclusions

We proposed extensions to the resistive bridging fault model which allow us to evaluate the impact of two advanced testing methods on the coverage of shorts. The methods investigated are very different in terms of their conceptual approach. Yet, both aim at increasing the coverage of resistive shorts beyond what may be obtained by “traditional” voltage testing.

In the first part of the chapter we addressed low-X testing, i.e. low-voltage testing, low-temperature testing, and their combination. We extended the electrical model of the resistive bridging fault model to account for the effect of both techniques on transistor behavior and defect resistance. Furthermore, we proposed advanced fault coverage metrics which reflect the coverage impact of low-X testing. The metrics accurately discriminate between shorts detectable under the nominal conditions, i.e. hard defects, and those defects which are detectable by low-X testing only, i.e. flaws. Based on the experimental results, we gave indications for optimal use of each individual technique and their combination. Moreover, we demonstrated that under certain situations, low-voltage testing can cause coverage loss with respect to the range of short resistances detectable under the nominal conditions. Experimental data underlines that this effect is actually observable for the benchmark circuits considered.

The second part of this chapter dealt with Delta-IDDQ testing. The extensions proposed there enable the evaluation of both voltage and quiescent current testing within the framework of the resistive bridging fault model. Again, fault coverage definitions were introduced which accurately discriminate between hard defects, detectable by voltage testing, and flaws, which are covered by Delta-IDDQ testing. Our experiments accounted for the specifics of Delta-IDDQ testing, which prevent the detection of bridging faults activated by every test vector (always activated faults). Experimental results confirm that these shorts have a very detrimental impact on the fault coverage attainable by Delta-IDDQ testing, and thus must be addressed appropriately during both test pattern generation and evaluation. We could demonstrate that Delta-IDDQ testing allows excellent coverage of resistive shorts, in particular if combined with voltage testing. Furthermore, we could show that the tight integration of voltage and current testing aids to correctly resolve the non-trivial detection conditions which may arise when applying Delta-IDDQ testing to sequential circuits.

Both extensions discussed in this chapter emphasize the advantages of the advanced analytical concept forming the basis of the resistive bridging fault model. Once the model has been extended to account for specific electrical and physical phenomena, their impact can be evaluated for several parameter configurations without any additional effort. Furthermore, the model’s parametric nature allows to compare conceptually different techniques, such as voltage and current testing, side by side.

Table 9.3: Non-nominal fault coverage FC^{nn} for 10 time units and $V_{\text{dd}}^{\text{nom}} = 3.3 \text{ V}$.

Circuit	3.3 V	3.0 V	2.8 V	2.5 V	2.0 V
c0432	50.89	51.09	43.00	41.89	32.68
c0499	61.75	62.76	62.26	62.29	51.75
c0880	81.10	79.76	80.05	79.71	76.74
c1355	71.79	73.36	72.70	72.10	60.25
c1908	81.47	81.32	80.01	77.25	69.84
c2670	68.46	67.67	67.29	64.79	55.72
c3540	64.18	62.16	57.84	53.72	47.06
c5315	69.57	70.62	63.09	62.17	56.52
c7552	75.83	75.82	73.81	71.44	63.06
cs00208	58.84	59.72	56.74	54.92	46.76
cs00298	86.72	86.28	85.56	82.94	77.61
cs00344	81.75	82.19	79.14	79.40	74.16
cs00349	81.88	82.23	79.21	79.47	74.37
cs00382	83.54	82.57	82.94	78.95	68.42
cs00386	45.45	46.26	46.56	46.63	29.79
cs00400	82.94	81.99	82.32	78.58	56.67
cs00420	58.33	58.55	57.87	57.16	53.11
cs00444	61.86	57.18	57.09	57.46	57.43
cs00510	74.54	72.87	72.77	71.44	61.04
cs00526	73.57	70.27	70.28	70.96	67.93
cs00641	90.30	89.10	88.16	82.98	68.39
cs00713	89.33	87.77	86.55	81.37	66.34
cs00820	39.10	35.98	34.23	31.44	22.50
cs00832	37.51	34.94	34.35	31.24	22.41
cs00838	53.62	52.20	52.11	51.08	46.87
cs00953	54.25	55.15	55.30	55.74	51.76
cs01196	41.02	39.83	39.47	38.27	33.33
cs01238	42.53	41.60	41.15	39.89	34.84
cs01423	71.17	69.54	69.61	65.36	62.15
cs01488	45.14	39.96	39.79	37.17	28.44
cs01494	44.40	39.95	39.72	37.12	27.93
cs05378	75.12	74.89	72.62	71.47	68.77
cs09234	64.16	64.44	63.68	61.47	55.00
cs13207	85.09	85.84	85.36	84.52	79.12
cs15850	76.29	75.82	75.18	74.17	68.61
cs35932	69.59	61.44	61.86	46.95	38.86
cs38417	83.96	84.94	84.53	83.10	76.72
cs38584	67.83	64.31	64.71	64.72	62.73
∅	66.97	65.85	64.71	62.67	55.15

Table 9.4: Non-nominal fault coverage FC^{nn} for 100 time units and $V_{dd}^{nom} = 3.3\text{ V}$.

Circuit	3.3 V	3.0 V	2.8 V	2.5 V	2.0 V
c0432	97.71	98.32	98.43	98.70	98.02
c0499	89.26	86.42	86.57	86.55	84.81
c0880	96.32	97.14	97.77	97.98	97.90
c1355	97.02	97.13	97.27	97.15	95.14
c1908	95.07	95.86	95.55	95.62	93.41
c2670	92.73	93.59	93.93	93.22	90.87
c3540	93.40	93.77	93.71	92.79	90.13
c5315	99.02	99.33	99.35	99.00	97.88
c7552	97.63	98.23	98.44	98.22	97.39
cs00208	94.66	93.01	93.26	92.50	90.71
cs00298	98.43	98.83	98.69	98.81	98.36
cs00344	99.22	99.28	99.39	99.36	98.36
cs00349	99.24	99.29	99.40	99.36	98.33
cs00382	99.09	99.40	99.49	99.35	98.71
cs00386	74.47	73.99	74.41	73.91	73.92
cs00400	99.09	99.40	99.33	99.31	98.63
cs00420	80.11	80.29	80.47	79.93	76.36
cs00444	98.30	98.74	98.77	98.88	97.56
cs00510	97.70	97.89	98.07	97.78	96.88
cs00526	96.11	96.88	96.15	96.64	95.66
cs00641	98.50	99.24	99.61	99.51	99.47
cs00713	98.63	99.26	99.51	99.36	99.33
cs00820	79.48	77.92	77.57	74.09	69.09
cs00832	78.60	77.17	76.69	73.33	67.88
cs00838	62.81	63.36	62.65	62.84	62.66
cs00953	86.87	87.17	87.25	87.11	85.81
cs01196	82.68	82.17	82.59	81.58	76.93
cs01238	83.74	83.42	83.93	83.25	79.30
cs01423	96.05	96.76	97.09	96.73	95.66
cs01488	84.49	83.85	83.57	82.12	77.59
cs01494	84.10	83.43	82.92	81.56	77.23
cs05378	93.02	93.62	93.70	93.54	92.52
cs09234	78.56	79.37	80.02	79.58	78.10
cs13207	93.33	94.49	95.06	95.27	94.82
cs15850	90.26	91.26	91.67	91.69	91.06
cs35932	99.32	99.26	98.94	98.68	97.68
cs38417	95.16	96.27	96.89	97.12	96.82
cs38584	88.24	88.90	89.12	89.21	88.40
∅	91.27	91.41	91.51	91.10	89.46

Table 9.5: Non-nominal fault coverage FC^{nn} for 1,000 time units and $V_{dd}^{nom} = 3.3$ V.

Circuit	3.3 V	3.0 V	2.8 V	2.5 V	2.0 V
c0432	99.78	99.90	99.96	100.00	99.94
c0499	99.99	100.00	100.00	100.00	99.95
c0880	98.88	99.26	99.47	99.74	99.90
c1355	99.89	99.93	99.92	99.90	99.80
c1908	99.25	99.52	99.53	99.46	99.37
c2670	95.93	96.62	96.90	96.97	96.84
c3540	99.19	99.43	99.41	99.41	99.11
c5315	99.96	99.99	100.00	100.00	100.00
c7552	99.12	99.46	99.57	99.60	99.59
cs00208	99.56	99.52	99.53	99.33	98.25
cs00298	99.98	99.99	99.99	100.00	100.00
cs00344	99.97	99.98	99.98	99.99	100.00
cs00349	99.97	99.98	99.98	99.99	100.00
cs00382	99.96	99.97	99.98	99.98	99.99
cs00386	99.82	99.46	99.46	99.49	98.95
cs00400	99.96	99.98	99.98	99.98	99.99
cs00420	85.20	85.63	85.82	86.02	85.96
cs00444	99.98	99.98	99.99	99.96	99.96
cs00510	99.98	99.99	99.99	99.99	99.85
cs00526	99.44	99.62	99.73	99.80	99.59
cs00641	99.61	99.84	99.96	99.92	99.92
cs00713	99.68	99.84	99.93	99.90	99.90
cs00820	96.29	96.49	96.60	96.64	95.99
cs00832	96.13	96.31	96.44	96.47	95.69
cs00838	69.83	70.29	70.53	70.76	70.27
cs00953	97.33	97.76	98.00	98.19	98.13
cs01196	96.07	96.57	96.78	96.71	95.92
cs01238	96.49	96.96	97.12	97.24	96.71
cs01423	99.21	99.49	99.65	99.80	99.75
cs01488	99.65	99.77	99.81	99.77	99.47
cs01494	99.66	99.77	99.82	99.71	99.51
cs05378	98.68	98.91	99.04	99.16	99.00
cs09234	90.92	91.37	91.61	91.23	89.06
cs13207	95.90	96.86	97.17	97.26	97.09
cs15850	96.76	97.39	97.40	97.52	97.20
cs35932	100.00	100.00	100.00	100.00	100.00
cs38417	97.73	98.48	98.77	98.94	98.87
cs38584	92.42	92.77	92.95	93.06	92.43
∅	97.32	97.55	97.65	97.68	97.42

Table 9.6: Flaw coverage $FC_{\text{flaw}}^{\text{nn}}$ for 1,000 time units.

Circuit	3.0 V	2.8 V	2.5 V	2.0 V
c0432	23.47	39.39	60.22	88.62
c0499	18.73	33.61	53.40	85.82
c0880	21.34	35.29	55.65	87.18
c1355	17.64	31.92	51.53	85.28
c1908	18.87	33.09	52.65	84.09
c2670	17.28	30.75	51.46	83.95
c3540	20.97	35.46	56.28	86.99
c5315	20.34	34.97	56.45	88.18
c7552	19.02	33.34	54.50	86.93
cs00208	22.80	37.32	58.67	86.65
cs00298	23.26	38.12	60.00	89.28
cs00344	22.00	36.00	57.65	88.15
cs00349	22.07	36.13	57.80	88.25
cs00382	24.82	40.32	62.43	90.44
cs00386	20.73	34.40	55.84	86.76
cs00400	24.65	40.11	62.12	90.26
cs00420	17.86	29.70	47.92	74.25
cs00444	23.59	38.97	60.19	89.25
cs00510	24.56	40.05	61.82	89.91
cs00526	23.87	38.64	60.08	88.40
cs00641	20.64	35.05	55.78	88.20
cs00713	20.74	35.18	55.85	88.03
cs00820	23.24	37.95	57.98	84.02
cs00832	23.33	38.08	58.10	83.93
cs00838	14.09	23.55	38.21	59.47
cs00953	21.48	35.73	56.85	86.31
cs01196	19.38	32.79	53.52	83.08
cs01238	19.54	33.13	53.99	83.62
cs01423	21.18	35.31	56.20	87.43
cs01488	21.38	35.25	57.27	88.09
cs01494	21.39	35.26	57.26	88.18
cs05378	25.67	40.78	62.80	90.42
cs09234	16.27	28.00	46.90	76.97
cs13207	17.86	30.68	51.04	84.77
cs15850	18.98	31.99	52.35	85.08
cs35932	19.46	33.51	53.73	85.84
cs38417	19.15	32.63	53.73	86.73
cs38584	19.12	32.01	51.57	80.22
∅	20.81	34.85	55.52	85.50

Table 9.7: Combined fault coverage $FC_{\text{comb}}^{\text{nn}}$ for $T^{\text{nom}} = 370 \text{ K}$, $T = 300 \text{ K}$, $V_{\text{dd}}^{\text{nom}} = 3.3 \text{ V}$ and different values of V_{dd} .

Circuit	G-FE	$FC_{\text{comb}}^{\text{nn}}$				
		3.3 V	3.0 V	2.8 V	2.5 V	2.0 V
c0432	99.84	99.88	99.91	99.94	99.98	100.00
c0499	99.99	100.00	100.00	100.00	100.00	100.00
c0880	98.52	99.10	99.33	99.51	99.74	99.91
c1355	99.90	99.94	99.96	99.97	99.99	100.00
c1908	99.08	99.33	99.43	99.51	99.60	99.69
c2670	92.96	93.88	94.14	94.22	94.39	94.62
c3540	99.25	99.47	99.54	99.57	99.62	99.66
c5315	99.88	99.93	99.96	99.98	99.99	100.00
c7552	97.99	98.39	98.50	98.55	98.64	98.76
cs00208	99.57	99.73	99.78	99.80	99.81	99.82
cs00298	99.99	100.00	100.00	100.00	100.00	100.00
cs00344	99.97	99.99	99.99	99.99	100.00	100.00
cs00349	99.97	99.99	99.99	99.99	100.00	100.00
cs00382	99.96	99.97	99.98	99.98	99.99	99.99
cs00386	99.78	99.90	99.93	99.93	99.94	99.94
cs00400	99.96	99.97	99.98	99.98	99.99	99.99
cs00420	83.84	84.68	84.97	85.09	85.26	85.41
cs00444	99.99	99.99	100.00	100.00	100.00	100.00
cs00510	99.97	99.98	99.99	99.99	99.99	100.00
cs00526	99.56	99.70	99.79	99.82	99.85	99.88
cs00641	99.56	99.86	99.91	99.94	99.97	99.97
cs00713	99.61	99.86	99.91	99.93	99.95	99.95
cs00820	95.88	96.36	96.66	96.74	96.85	96.97
cs00832	95.48	95.92	96.21	96.30	96.41	96.54
cs00838	68.10	68.79	69.02	69.14	69.36	69.59
cs00953	97.24	97.65	97.84	97.95	98.09	98.20
cs01196	95.17	96.14	96.50	96.66	96.83	97.00
cs01238	95.29	96.19	96.56	96.72	96.92	97.12
cs01423	98.92	99.42	99.59	99.65	99.73	99.77
cs01488	99.62	99.79	99.84	99.85	99.85	99.85
cs01494	99.66	99.80	99.84	99.85	99.85	99.85
cs05378	98.57	99.02	99.24	99.31	99.37	99.39
cs09234	89.77	91.14	91.58	91.75	91.92	92.05
cs13207	94.15	95.34	95.76	95.93	96.11	96.27
cs15850	94.77	95.59	95.90	96.05	96.20	96.32
cs35932	100.00	100.00	100.00	100.00	100.00	100.00
cs38417	95.53	96.39	96.71	96.88	97.06	97.20
cs38584	92.15	92.77	93.02	93.16	93.34	93.51
∅	96.83	97.21	97.35	97.41	97.49	97.56

Table 9.8: Flaw coverage $FC_{\text{flaw}}^{\text{nn}}$ for $V_{\text{dd}}^{\text{nom}} = 3.3 \text{ V}$, $T^{\text{nom}} = T = 370 \text{ K}$ and different values of V_{dd} .

Circuit	3.0 V	2.8 V	2.5 V	2.0 V
c0432	51.55	68.35	83.41	95.61
c0499	34.90	50.90	68.16	88.70
c0880	33.07	48.06	64.50	84.15
c1355	49.23	69.70	87.21	97.63
c1908	34.56	50.70	67.96	86.28
c2670	28.25	40.77	57.36	79.51
c3540	32.90	46.15	63.71	86.57
c5315	34.85	49.68	67.80	89.63
c7552	34.13	49.93	67.83	87.63
cs00208	27.91	42.60	62.10	86.80
cs00298	26.30	38.19	55.72	83.16
cs00344	25.07	40.15	60.67	86.37
cs00349	25.32	40.38	61.10	86.66
cs00382	37.94	54.10	72.98	92.11
cs00386	18.03	28.57	48.37	80.84
cs00400	38.15	54.48	73.25	92.14
cs00420	21.08	32.15	47.91	70.25
cs00444	40.28	57.76	75.89	93.48
cs00510	29.91	44.63	64.45	89.41
cs00526	27.13	38.50	58.59	84.30
cs00641	18.96	28.75	49.33	80.91
cs00713	19.33	29.66	50.09	81.23
cs00820	27.13	37.41	55.24	79.77
cs00832	27.12	37.06	54.77	79.34
cs00838	16.80	25.54	38.06	56.49
cs00953	28.29	43.18	62.89	85.23
cs01196	28.10	40.60	57.86	81.23
cs01238	29.46	42.08	58.77	81.18
cs01423	25.66	40.09	59.51	84.81
cs01488	23.23	35.34	54.93	84.21
cs01494	23.53	35.66	55.18	84.34
cs05378	32.85	46.65	66.27	88.88
cs09234	18.76	28.87	46.54	73.02
cs13207	20.71	31.46	50.30	77.14
cs15850	21.58	33.12	52.30	78.75
cs35932	36.49	53.72	71.81	87.32
cs38417	22.63	34.88	53.67	79.61
cs38584	23.63	36.25	53.34	76.40
∅	28.81	42.27	60.52	83.71

Table 9.9: Flaw coverage $FC_{\text{flaw}}^{\text{nn}}$ for $V_{\text{dd}}^{\text{nom}} = 3.3 \text{ V}$, $T^{\text{nom}} = 370 \text{ K}$, $T = 300 \text{ K}$ and different values of V_{dd} .

Circuit	3.3 V	3.0 V	2.8 V	2.5 V	2.0 V
c0432	5.94	33.24	54.55	80.78	97.17
c0499	11.50	13.51	32.71	68.63	94.73
c0880	14.90	32.43	50.35	72.64	92.72
c1355	1.63	11.13	37.46	76.03	97.44
c1908	12.34	23.19	43.51	71.38	92.03
c2670	17.37	30.29	45.50	67.77	87.16
c3540	23.69	41.16	55.87	76.29	94.88
c5315	20.78	34.16	51.14	76.27	96.43
c7552	15.66	28.16	46.79	73.42	93.45
cs00208	23.99	42.36	57.33	76.83	95.18
cs00298	24.08	40.57	54.20	74.24	94.87
cs00344	26.16	43.30	57.27	76.66	95.38
cs00349	25.69	42.84	56.92	76.55	95.50
cs00382	18.12	39.17	55.50	78.37	96.91
cs00386	30.36	44.03	55.72	73.25	94.29
cs00400	16.69	37.19	54.50	77.75	96.76
cs00420	18.52	33.71	45.74	61.82	79.00
cs00444	12.78	29.39	49.44	77.20	96.78
cs00510	17.51	34.65	52.21	75.38	96.04
cs00526	21.06	39.46	52.12	73.56	93.70
cs00641	31.13	46.83	58.50	74.76	95.36
cs00713	30.62	45.83	57.91	74.66	95.18
cs00820	16.93	36.82	48.26	67.34	87.94
cs00832	16.44	35.98	47.32	66.60	87.54
cs00838	14.41	26.50	36.11	48.99	62.81
cs00953	14.91	30.52	47.38	69.94	92.14
cs01196	16.93	33.28	48.67	69.50	89.70
cs01238	16.04	33.09	48.72	69.44	89.16
cs01423	22.57	40.86	55.43	74.74	94.68
cs01488	31.26	49.67	60.58	76.79	95.49
cs01494	31.40	49.87	60.79	76.93	95.54
cs05378	23.79	52.74	64.75	80.59	96.41
cs09234	21.17	34.71	46.69	64.60	84.76
cs13207	22.07	35.75	48.64	68.00	89.10
cs15850	23.91	38.68	51.29	70.09	90.18
cs35932	9.82	18.20	37.53	68.96	89.82
cs38417	23.60	39.83	52.43	70.89	90.68
cs38584	20.42	36.78	49.40	67.28	85.71
∅	19.67	35.79	50.77	72.23	91.91

Table 9.10: Flaw coverage $FC_{\text{flaw}}^{\text{nn}}$ for $V_{\text{dd}}^{\text{nom}} = 3.3 \text{ V}$, $T^{\text{nom}} = 370 \text{ K}$, $T = 196 \text{ K}$ and different values of V_{dd} .

Circuit	3.3 V	3.0 V	2.8 V	2.5 V	2.0 V
c0432	24.34	29.69	43.50	70.35	90.49
c0499	28.52	30.97	38.62	48.64	72.38
c0880	40.70	47.98	56.21	77.17	97.61
c1355	4.63	5.65	8.73	29.53	54.52
c1908	29.92	34.91	42.55	60.52	92.21
c2670	45.01	52.07	58.78	70.67	88.97
c3540	57.53	65.30	71.74	81.54	96.92
c5315	49.55	56.43	63.76	76.90	98.68
c7552	39.35	45.47	53.36	69.26	94.79
cs00208	63.06	72.78	78.24	87.01	95.78
cs00298	64.11	70.76	76.89	90.34	98.96
cs00344	66.65	75.78	80.46	86.59	97.32
cs00349	66.37	75.65	80.64	86.99	97.13
cs00382	49.31	57.99	64.22	79.00	96.63
cs00386	76.28	84.32	89.96	95.65	98.69
cs00400	46.79	55.24	61.89	78.05	95.93
cs00420	51.60	60.62	65.73	73.34	81.59
cs00444	35.64	42.40	48.82	63.13	90.18
cs00510	51.74	60.30	68.27	82.87	95.30
cs00526	58.93	67.45	73.18	84.28	96.57
cs00641	75.30	84.15	90.28	96.30	99.68
cs00713	73.98	82.59	88.50	94.30	99.34
cs00820	49.10	57.81	63.94	77.45	89.93
cs00832	47.81	56.53	62.75	76.39	89.73
cs00838	40.41	47.49	51.53	57.58	64.42
cs00953	44.30	54.01	60.74	73.83	86.08
cs01196	47.76	56.13	62.86	77.20	92.60
cs01238	45.91	54.24	61.08	76.04	91.68
cs01423	60.18	70.63	78.91	89.77	96.99
cs01488	77.00	86.58	91.55	96.68	99.35
cs01494	77.10	86.68	91.61	96.71	99.37
cs05378	68.01	84.50	90.60	96.50	99.24
cs09234	55.07	63.71	69.82	77.38	88.96
cs13207	56.27	64.98	72.09	80.25	93.60
cs15850	59.09	68.03	74.69	82.55	94.13
cs35932	23.83	27.55	33.42	46.37	73.08
cs38417	59.93	69.57	75.51	82.85	95.32
cs38584	53.29	62.25	68.62	77.96	88.00
∅	51.69	59.72	66.16	77.58	91.37

Table 9.11: Flaw coverage $FC_{\text{flaw}}^{\text{nn}}$ for $V_{\text{dd}}^{\text{nom}} = 3.3 \text{ V}$, $T^{\text{nom}} = 300 \text{ K}$, $T = 196 \text{ K}$ and different values of V_{dd} .

Circuit	3.3 V	3.0 V	2.8 V	2.5 V	2.0 V
c0432	21.50	30.66	46.20	72.66	90.24
c0499	22.59	29.52	36.87	54.63	71.93
c0880	31.13	42.51	54.23	79.06	97.37
c1355	4.33	7.19	10.89	36.12	54.06
c1908	23.33	33.50	41.74	68.53	91.94
c2670	33.47	46.14	54.73	72.75	88.49
c3540	43.69	57.46	65.87	82.24	96.50
c5315	37.88	51.18	59.91	80.33	98.16
c7552	30.33	42.53	51.17	74.34	94.23
cs00208	47.10	61.39	70.19	84.68	95.42
cs00298	48.42	59.05	69.43	87.15	98.75
cs00344	50.02	63.48	71.38	84.70	96.93
cs00349	49.98	63.72	71.86	84.93	96.74
cs00382	37.77	52.51	62.48	81.63	96.25
cs00386	56.83	69.53	78.81	89.86	98.29
cs00400	36.15	50.78	60.98	80.75	95.54
cs00420	37.83	50.49	58.32	71.09	81.26
cs00444	27.87	38.98	47.07	69.95	89.75
cs00510	40.00	52.32	63.47	81.55	95.01
cs00526	44.59	57.07	66.40	82.32	96.78
cs00641	55.36	69.85	79.40	90.45	99.24
cs00713	54.47	68.51	77.81	89.24	98.90
cs00820	37.18	48.83	59.44	75.50	90.11
cs00832	36.29	47.77	58.39	74.73	90.15
cs00838	29.49	39.42	45.66	55.95	64.18
cs00953	33.51	47.14	56.74	72.56	86.34
cs01196	35.63	48.42	58.52	77.41	92.21
cs01238	34.34	46.97	57.17	76.49	91.32
cs01423	45.25	61.47	72.00	86.08	96.62
cs01488	57.14	72.31	80.86	91.23	98.97
cs01494	57.23	72.44	80.94	91.28	99.00
cs05378	49.83	71.74	81.06	91.77	98.94
cs09234	39.64	53.44	61.99	76.06	88.53
cs13207	40.96	55.74	64.59	79.60	93.14
cs15850	43.25	58.00	66.85	81.20	93.69
cs35932	18.92	26.96	32.92	53.41	72.51
cs38417	43.77	58.57	67.34	81.55	94.88
cs38584	39.57	52.94	61.97	76.23	87.64
∅	38.86	51.59	60.68	77.37	91.05

Table 9.12: Fault coverages for Delta-IDDQ testing with 1,000 test vectors for voltage testing, current threshold $\Delta I_{\text{limit}} = 100 \mu\text{A}$ and 10 IDDQ measurements.

Circuit	G -FE	Delta-IDDQ				IDDQ	
		AA	FC^{IDDQ}	$\text{FC}^{\text{IDDQ}}_{\text{comb}}$	$\text{FC}^{\text{IDDQ}}_{\text{flaw}}$	FC^{IDDQ}	$\text{FC}^{\text{IDDQ}}_{\text{comb}}$
c0432	99.78	53	97.60	100.00	97.58	98.62	100.00
c0499	99.99	2	92.81	100.00	92.03	92.84	100.00
c0880	98.88	167	96.27	99.91	96.18	97.94	99.94
c1355	99.89	170	95.99	99.93	95.93	97.70	99.93
c1908	99.25	88	98.07	99.93	96.79	98.97	99.93
c2670	95.93	120	97.41	99.93	97.31	98.61	99.96
c3540	99.19	172	95.58	99.91	95.13	97.31	99.93
c5315	99.96	34	99.21	100.00	99.15	99.55	100.00
c7552	99.12	39	99.08	99.98	99.00	99.47	99.98
cs00208	99.56	370	81.82	99.95	81.33	91.22	99.96
cs00298	99.98	42	98.31	100.00	97.83	99.26	100.00
cs00344	99.97	31	99.09	100.00	98.74	99.49	100.00
cs00349	99.97	37	99.01	100.00	98.65	99.49	100.00
cs00382	99.96	112	97.25	99.98	97.11	98.69	99.98
cs00386	99.82	226	91.60	99.97	91.61	94.04	99.97
cs00400	99.96	134	96.75	99.98	96.61	98.37	99.98
cs00420	85.20	896	82.17	96.14	81.87	91.20	96.21
cs00444	99.98	108	97.22	100.00	97.04	98.30	100.00
cs00510	99.98	371	92.80	100.00	92.78	96.52	100.00
cs00526	99.44	177	96.50	99.94	96.18	98.29	99.98
cs00641	99.61	80	98.22	99.97	98.02	99.02	99.98
cs00713	99.68	121	97.62	99.95	97.28	98.84	99.95
cs00820	96.29	334	93.99	99.77	93.75	97.37	99.86
cs00832	96.13	330	93.46	99.70	93.29	96.80	99.79
cs00838	69.83	1,342	73.12	90.19	72.80	86.62	90.43
cs00953	97.33	1,942	61.44	99.21	61.41	80.89	99.35
cs01196	96.07	486	88.88	99.06	88.88	93.75	99.19
cs01238	96.49	494	88.00	98.67	87.97	92.94	98.90
cs01423	99.21	77	97.57	99.94	97.06	98.35	99.96
cs01488	99.65	81	93.61	99.96	93.58	94.43	99.96
cs01494	99.66	115	92.98	99.96	92.92	94.14	99.97
cs05378	98.68	172	95.89	99.83	95.84	97.61	99.89
cs09234	90.92	188	95.98	99.24	95.88	97.87	99.50
cs13207	95.90	190	95.63	99.17	95.57	97.51	99.28
cs15850	96.76	161	96.33	99.58	96.29	97.94	99.76
cs35932	100.00	66	98.89	100.00	98.46	99.56	100.00
cs38417	97.73	92	98.09	99.84	98.08	99.01	99.90
cs38584	92.42	229	95.57	98.95	94.77	97.88	99.60
\emptyset	97.32		93.68	99.44	93.44	96.48	99.50

Table 9.13: Fault coverages for Delta-IDDQ testing with 1,000 test vectors for voltage testing, current threshold $\Delta I_{\text{limit}} = 100 \mu\text{A}$ and 100 IDDQ measurements.

Circuit	G -FE	Delta-IDDQ				IDDQ	
		AA	FC^{IDDQ}	$\text{FC}^{\text{IDDQ}}_{\text{comb}}$	$\text{FC}^{\text{IDDQ}}_{\text{flaw}}$	FC^{IDDQ}	$\text{FC}^{\text{IDDQ}}_{\text{comb}}$
c0432	99.78	0	100.00	100.00	99.96	100.00	100.00
c0499	99.99	0	96.70	100.00	95.86	96.70	100.00
c0880	98.88	8	99.82	99.96	99.73	99.90	99.96
c1355	99.89	89	97.92	99.94	97.85	98.81	99.94
c1908	99.25	45	99.30	99.97	98.00	99.76	99.97
c2670	95.93	6	99.94	100.00	99.87	100.00	100.00
c3540	99.19	26	99.71	100.00	99.24	99.97	100.00
c5315	99.96	5	99.95	100.00	99.89	100.00	100.00
c7552	99.12	1	99.98	100.00	99.90	99.99	100.00
cs00208	99.56	20	99.26	100.00	98.70	99.77	100.00
cs00298	99.98	8	99.80	100.00	99.31	99.98	100.00
cs00344	99.97	10	99.87	100.00	99.51	100.00	100.00
cs00349	99.97	10	99.87	100.00	99.52	100.00	100.00
cs00382	99.96	11	99.85	99.99	99.69	99.99	99.99
cs00386	99.82	26	99.43	100.00	99.37	99.71	100.00
cs00400	99.96	13	99.83	99.99	99.69	99.99	99.99
cs00420	85.20	368	92.75	97.01	92.47	96.45	97.03
cs00444	99.98	14	99.85	100.00	99.72	99.99	100.00
cs00510	99.98	2	99.97	100.00	99.97	99.99	100.00
cs00526	99.44	13	99.86	100.00	99.59	99.99	100.00
cs00641	99.61	14	99.84	100.00	99.64	99.98	100.00
cs00713	99.68	27	99.65	99.97	99.31	99.92	99.97
cs00820	96.29	13	99.83	99.98	99.68	99.96	99.98
cs00832	96.13	14	99.79	99.96	99.62	99.93	99.96
cs00838	69.83	876	83.43	93.13	83.14	92.25	93.18
cs00953	97.33	272	94.62	99.65	94.62	97.35	99.68
cs01196	96.07	18	99.56	99.89	99.53	99.74	99.89
cs01238	96.49	20	99.44	99.87	99.44	99.64	99.87
cs01423	99.21	2	99.91	99.99	99.38	99.93	99.99
cs01488	99.65	2	99.92	100.00	99.92	99.94	100.00
cs01494	99.66	3	99.95	100.00	99.90	99.98	100.00
cs05378	98.68	20	99.63	100.00	99.58	99.83	100.00
cs09234	90.92	66	98.67	99.62	98.57	99.34	99.68
cs13207	95.90	66	98.47	99.43	98.41	99.11	99.48
cs15850	96.76	27	99.23	99.90	99.20	99.50	99.92
cs35932	100.00	4	99.96	100.00	99.55	100.00	100.00
cs38417	97.73	6	99.83	99.96	99.82	99.89	99.96
cs38584	92.42	64	99.07	99.52	98.24	99.72	99.79
\emptyset	97.32		98.80	99.68	98.56	99.39	99.69

Table 9.14: Fault coverages for Delta-IDDQ testing with 1,000 test vectors for voltage testing, current threshold $\Delta I_{\text{limit}} = 100 \mu\text{A}$ and 1,000 IDDQ measurements.

Circuit	G-FE	Delta-IDDQ				IDDQ	
		AA	FC ^{IDDQ}	FC ^{IDDQ} _{comb}	FC ^{IDDQ} _{flaw}	FC ^{IDDQ}	FC ^{IDDQ} _{comb}
c0432	99.78	0	100.00	100.00	99.96	100.00	100.00
c0499	99.99	0	100.00	100.00	99.11	100.00	100.00
c0880	98.88	0	99.98	99.98	99.89	99.98	99.98
c1355	99.89	0	100.00	100.00	99.93	100.00	100.00
c1908	99.25	27	99.71	99.98	98.40	99.98	99.98
c2670	95.93	5	99.95	100.00	99.88	100.00	100.00
c3540	99.19	25	99.75	100.00	99.28	100.00	100.00
c5315	99.96	5	99.95	100.00	99.89	100.00	100.00
c7552	99.12	1	99.99	100.00	99.91	100.00	100.00
cs00208	99.56	6	99.85	100.00	99.27	100.00	100.00
cs00298	99.98	7	99.84	100.00	99.35	100.00	100.00
cs00344	99.97	10	99.87	100.00	99.51	100.00	100.00
cs00349	99.97	10	99.87	100.00	99.52	100.00	100.00
cs00382	99.96	11	99.86	100.00	99.71	100.00	100.00
cs00386	99.82	0	100.00	100.00	99.94	100.00	100.00
cs00400	99.96	13	99.83	99.99	99.69	99.99	99.99
cs00420	85.20	244	95.27	97.71	95.01	97.72	97.72
cs00444	99.98	14	99.86	100.00	99.73	100.00	100.00
cs00510	99.98	0	100.00	100.00	100.00	100.00	100.00
cs00526	99.44	12	99.88	100.00	99.61	100.00	100.00
cs00641	99.61	14	99.86	100.00	99.66	100.00	100.00
cs00713	99.68	19	99.81	100.00	99.47	100.00	100.00
cs00820	96.29	9	99.91	100.00	99.76	100.00	100.00
cs00832	96.13	8	99.92	100.00	99.75	100.00	100.00
cs00838	69.83	702	86.69	93.72	86.41	93.76	93.76
cs00953	97.33	17	99.65	99.81	99.65	99.82	99.82
cs01196	96.07	1	99.99	100.00	99.96	100.00	100.00
cs01238	96.49	1	99.99	100.00	99.99	100.00	100.00
cs01423	99.21	0	100.00	100.00	99.47	100.00	100.00
cs01488	99.65	0	100.00	100.00	100.00	100.00	100.00
cs01494	99.66	0	100.00	100.00	99.95	100.00	100.00
cs05378	98.68	3	99.97	100.00	99.92	100.00	100.00
cs09234	90.92	19	99.67	99.82	99.57	99.86	99.86
cs13207	95.90	16	99.43	99.57	99.39	99.59	99.59
cs15850	96.76	1	99.96	99.97	99.93	99.97	99.97
cs35932	100.00	4	99.96	100.00	99.55	100.00	100.00
cs38417	97.73	2	99.96	99.98	99.95	99.98	99.98
cs38584	92.42	33	99.56	99.74	98.76	99.89	99.89
∅	97.32		99.42	99.74	99.18	99.75	99.75

10 Benchmarking BIST Techniques

Design for testability (DFT) methods are commonly used to improve the testability of devices. In many cases, however, improving the testability primarily means achieving a dedicated single-stuck-at fault coverage target. Even though in some applications this might already be very challenging, high single-stuck-at fault coverage cannot be sufficient. Particularly, as it is well-known that the single-stuck-at fault model is an imperfect representative of actual defects. Consequently, it is critical to question DFT measures with respect to their impact on the coverage of actual defects. Unfortunately, there is no such thing as a model for defects. Therefore, we propose to use the resistive bridging fault model as a surrogate for non-target defects [79], i.e. defects which have not explicitly been targeted during construction of the DFT hardware. This approach is supported by the fact that resistive shorts, modeled by the resistive bridging fault model, are a major defect contributor in nanoscale technologies. Furthermore, the model accurately reflects non-trivial electrical phenomena such as pattern-dependency and Byzantine-behavior – effects which are discounted by the stuck-at fault model.

In this chapter we will evaluate two techniques used in the context of built-in self test (BIST) in terms of their effect on the coverage of resistive shorts. During the past few years, BIST solutions have gained widespread use in practice to complement or even replace automatic test equipment (ATE) based test application. Obviously, it is important that these techniques do not only guarantee high single-stuck-at fault coverage, but actually provide a sufficient detection of defects. The techniques addressed in this chapter use several measures to achieve excellent single-stuck-at fault coverage. Yet, their architectures do not consider resistive bridging faults, and thus ignore many characteristics of shorts. As a consequence, sufficient coverage of shorts cannot be presumed and may even be called purely accidental. Hence, it is likely that the coverage of resistive bridging faults is a valid indicator for the impact of a BIST solution on the detection of non-target defects.

The following Chapter 10.1 will target a BIST solution addressing the input side of the circuit under test (CUT). The technique selectively modifies pseudo-random test patterns supplied by an on-chip test pattern source to increase their single-stuck-at fault coverage. Subsequently, we will move our focus to the output side of the CUT. Many circuits produce unknown values which may impair the correct operation of a test response evaluator (TRE). A technique which prevents the unknown values from entering the response evaluation hardware is explored in Chapter 10.2. Conclusions are given in Chapter 10.3.

10.1 Detection of Non-Target Defects by Deterministic Logic BIST

Usually, the pattern generator (PG) used in BIST solutions generates pseudo-random patterns. Unfortunately, in many circuits a non-negligible share of faults is random pattern resistant [40]. This means that these faults may only be detected by very specific input combinations. While patterns detecting some of the random pattern resistant faults may be contained “by chance” in the pseudo-random sequence, it is very unlikely that the sequence covers all hard to detect faults. This problem may be alleviated to a certain degree by using extremely long pattern sequences which, however, is unacceptable in many applications. An alternative to this are deterministic patterns, specifically generated by an ATPG system for those faults not detected by the random pattern sequence. These deterministic patterns may be applied in addition to the pseudo-random sequence. This, however, also results in prolonged test application time. Further solutions include the insertion of test points [68] to enhance the observability of fault effects, weighted random testing [199, 205] which modifies the signal probabilities of the pseudo-random pattern sequence, and the Circular Self-Test Path technique [90, 135] that uses the circuit functionality to generate new test patterns.

Deterministic logic BIST (DLBIST) techniques directly modify the pseudo-random pattern sequence supplied by the PG to increase the fault coverage. Pseudo-random pattern generators used in *reseeding architectures* include linear feedback shift registers (LFSR) [91, 212], multiple-polynomial LFSRs [70, 71], twisted-ring counters [23], and folding counters [69]. These techniques omit those parts of the pseudo-random sequence which do not contribute to fault detection. A similar solution for two-pattern testing is based on a multiple input shift register [137, 207]. Alternatively, *pattern embedding* modifies some of the bits of a pseudo-random pattern sequence such that every vector from a given set of deterministic test patterns is contained in the sequence. Pattern embedding may be performed by either *bit-fixing* [187] or *bit-flipping* [86, 206]. Primarily, the techniques differ in the logic employed to alter the pseudo-random pattern sequence: While bit-fixing uses a combination of AND and OR gates, bit-flipping is restricted to XOR gates. A related technique is *Embedded Deterministic Test* (EDT) [141], which involves interaction with the tester and is therefore considered a test compression rather than a BIST method.

Commonly, DLBIST solutions are optimized for minimal silicon area, short test application time (given by the length of the pattern sequence), and high single-stuck-at fault coverage. Unfortunately the detection of actual defects is not considered. Since the correlation between detection of single-stuck-at faults and defect coverage is rather limited, it is questionable if DLBIST may achieve a sufficient coverage of non-target defects.

In the following, we present our study [W7, W8] on the relation between area cost, test length, and the coverage of non-target defects for a DLBIST scheme based on bit-flipping. We use resistive bridging faults as a surrogate for non-target defects. We synthesized several bit-flipping DLBIST circuitries for various benchmark circuits and pattern sequence lengths. In particular, we focused on the tradeoff between silicon area, sequence length,

and resistive bridging fault coverage. Our experimental results demonstrate that longer test sequences achieve higher coverage of non-target defects (resistive shorts) and imply a smaller bit-flipping logic. In particular, this dependency on the sequence length indicates that high single-stuck-at fault coverage alone is not sufficient to cover resistive bridging faults and that pseudo-random patterns are required as well. As both deterministic single-stuck-at and pseudo-random patterns are produced by bit-flipping DLBIST, we conclude, that this solution is well suited to cover non-target defects.

Chapter 10.1.1 will give a brief overview over bit-flipping DLBIST. Experimental results can be found in Chapter 10.1.2. In Chapter 10.3 conclusions are given.

10.1.1 Deterministic Logic BIST with Bit-Flipping

In bit-flipping the pseudo-random pattern sequence is modified by a special combinational logic, called *bit-flipping logic* (BFL), such that some of the patterns match the vectors contained in a given set of deterministic test patterns. The technique exploits the fact that pseudo-random sequences contain many *useless* test vectors (see e.g. [186]). These are patterns that do not contribute to the fault coverage in the sense that they do only detect faults which have already been covered by other patterns preceding them in the sequence. Furthermore, it is exploited that deterministic patterns contain a large number of don't care values (see e.g. [88]), i.e. bits which may be set to either the logical value 0 or 1. Therefore, a deterministic pattern may be mapped to a useless pseudo-random pattern by modifying just a few bits without sacrificing the advantages of the pseudo-random sequence.

Pattern embedding is performed in a preprocessing step. It identifies for each pattern d from a given set of deterministic patterns a useless pattern r , such that d can be mapped efficiently to r . The logic operations required to perform that mapping are captured by the *bit flipping function* (BFF). Once the given test set has been processed completely, the BFFs required to embed all deterministic patterns into the pseudo-random pattern sequence are combined to form the control logic of the BFL.

In the next section, we will first introduce the architecture of the DLBIST bit-flipping logic. Subsequently, we will give a short overview over the pattern embedding procedure which is required to understand the experimental results. More details on the bit-flipping logic and its synthesis may be found in [53, 86, 206].

Architecture of Deterministic Logic BIST

Figure 10.1 depicts the architecture used for deterministic logic BIST. The circuit under test is augmented by a BIST control unit which operates the source of the random patterns and

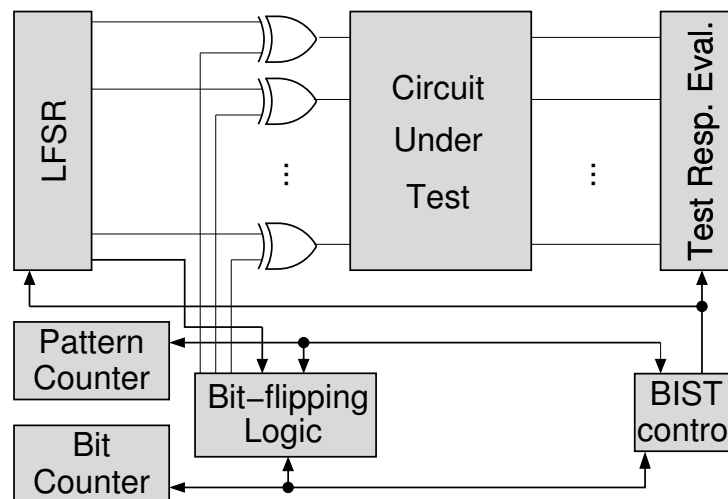


Figure 10.1: Schematic of circuit under test with bit-flipping logic.

the test response evaluator. In our case, the test patterns are supplied by an LFSR¹, while the outputs of the CUT are analyzed by a test response evaluator (e.g. a multiple-input shift register). The BIST control is supported by the *pattern counter* which keeps track of the patterns applied so far and the *bit counter* which records the number of bits shifted into the scan-chain(s). Both counters and the state of the LFSR are fed into the bit-flipping logic. The BFL controls which bits supplied by the LFSR are flipped by driving either the logical value 1 or 0 at the inputs of the XOR gates linking the LFSR with the CUT. The logical value 1 causes the bit to be flipped, while the logical value 0 leaves the bit unaltered. The architecture from Figure 10.1 may be applied to combinational circuits as well as to circuits containing one or multiple scan-chains [86].

Synthesis of Bit-Flipping Logic

The bit-flipping logic has three inputs: the state LF of the LFSR, the value of the pattern counter PC , and that of the bit counter BC . The bit flipping function implemented by the BFL maps each state of the DLBIST logic, defined as $LF \times PC \times BC$, to a set of logical values driven at the inputs of the XOR gates. For each bit i this assignment is derived as follows: Assume the deterministic ATPG pattern d is to be mapped to pseudo-random pattern r (supplied by the LFSR). When comparing the i -th bit position d_i and r_i of pattern d and r , respectively, three relevant cases may occur:

Matching bits: Bit d_i is specified and $r_i = d_i$, i.e. either $r_i = d_i = 1$ or $r_i = d_i = 0$.

Conflicting bits: Bit d_i is specified and $r_i \neq d_i$, i.e. $r_i = 1$ and $d_i = 0$ or $r_i = 0$ and $d_i = 1$.

Don't care bits: Bit d_i is unspecified and r_i has an arbitrary value.

¹Note that the correlation of the patterns produced by an LFSR may be reduced by a phase shifter. For bit-flipping, however, this is only optional as the same task may also be performed by the bit-flipping logic itself [86].

Conflicting bits must be flipped (BFL maps to 1) while matching bits must not be flipped (BFL maps to 0). It is irrelevant whether don't care bits are flipped or not. In the latter case, the BFL may either drive the logical value 0 or 1, depending on which option leads to a more efficient implementation of the BFF.

Five steps are required to derive the bit flipping function for a given combination of LFSR, CUT, and fault list:

- (1) Run fault simulation of the LFSR sequence and drop detected faults.
- (2) Perform ATPG (without random filling of the don't care positions) for the remaining faults.
- (3) For every deterministic test pattern obtained by ATPG, select a useless pseudo-random pattern from the pseudo-random sequence to be transformed into that pattern by flipping of individual bits. Determine the BFF necessary to perform the pattern conversion.
- (4) All BFFs are synthesized and a compact BFL is obtained that performs the transformation of useless patterns into deterministic patterns.
- (5) Perform fault simulation of the final sequence produced by the LFSR in combination with the BFL.

Steps (1) and (2) determine which faults are detected by the pseudo-random pattern sequence and generate test patterns which cover the remaining undetected faults. In Step (3), a useless pseudo-random pattern is assigned to each deterministic pattern. In order to obtain an efficient implementation of the BFL, the pseudo-random pattern with the minimum number of conflicting bits is selected. If several such patterns exist, the pattern is chosen which minimizes the number of:

- a) scan chains containing both matching and conflicting bits,
- b) clock cycles during which matching and conflicting bits are shifted into the scan chains.

Constraint a) attempts to minimize the BFL size per scan chain. Constraint b) has a lower priority and attempts to maximize logic sharing among the BFLs corresponding to different scan chains.

Once all deterministic patterns have been mapped to a pattern from the pseudo-random sequence, the BFL is generated in Step (4). The problem of BFL construction is formulated as an instance of logic synthesis with don't cares [17] which may be solved using a procedure based on binary decision diagrams [20, 39] (BDDs). All states of $LF \times PC \times BC$ which are mapped to the logical value 1 are represented as one BDD. Moreover, all states which are mapped to the logical value 0 are represented as a second BDD. These BDDs are transformed into an RTL description and synthesized using a commercial synthesis tool (see [53] for details). Finally in Step (5) fault simulation of the test pattern sequence which was produced by the resulting BFL is performed.

Table 10.1: Stuck-at coverage of pseudo-random sequences before deterministic pattern embedding.

Circuit	Sequence length		
	1,000	5,000	10,000
c7552	92.38	93.51	94.68
cs09234	72.31	80.79	83.60
cs13207	76.56	86.76	91.47
cs15850	84.58	89.98	91.14
cs38417	86.23	90.57	92.61
cs38584	90.47	93.44	94.31
\emptyset	83.76	89.18	91.30

10.1.2 Experimental Results

We applied the pattern embedding procedure to the ISCAS 85 [19] and the combinational cores of the ISCAS 89 [18] benchmark circuits. For each circuit pseudo-random test sequences of 1,000, 5,000, and 10,000 patterns were generated. For a large share of the considered circuits, the sequence of 10,000 patterns detected all detectable single-stuck-at faults. These circuits were excluded from consideration. For the remaining circuits, single-stuck-at fault coverages of the pseudo-random sequences can be found in Table 10.1 (the last row of the table gives average coverages). The faults which remained undetected by the sequences were targeted by a deterministic single-stuck-at ATPG which created test patterns with don't cares. Note that after embedding these patterns into the sequences, all non-redundant faults not aborted by the ATPG are detected.

We performed resistive bridging fault simulations for the pseudo-random pattern sequences before and after pattern embedding. The fault list contained 10,000 randomly selected non-feedback bridging faults, where available.² Alternatively, we could also use layout extracted bridging faults (refer to Chapter 3.3). We employed the Shockley technology model and parameters from the SPICE model card of a 0.35 μm technology from austriamicrosystems AG (AMS) to determine critical resistances (refer again to Chapter 5.2.2). When calculating fault coverages, we used the distribution ρ proposed in [165] which is based on data from [158]. Yet, the experiments in this chapter could be easily repeated with any other short resistance distribution. For each circuit, G -ADI was obtained by RBF-ATPG covered in Chapter 8. All experiments were performed on a 2 GHz Pentium IV with 2 GB RAM. The embedding procedure considers only single-stuck-at fault detection; hence resistive bridging faults are a valid surrogate for non-target defects.

Experimental results can be found in Table 10.2 for all circuits for which the sequence of 10,000 patterns did not detect all detectable single-stuck-at faults. The first column of the table states the name of the circuit. Then for each of the three sequence lengths under

²Note that unless otherwise specified the fault lists used in this chapter are equal to those employed in all other experiments discussed in Part II of this work.

Table 10.2: Fault coverage G -FE and logic size in GE for sequences of 1,000, 5,000, and 10,000 patterns.

Circuit	1,000 Patterns			5,000 Patterns			10,000 Patterns		
	G -FE		Emb'd	G -FE		Emb'd	G -FE		Emb'd
	Random	Emb'd	LSIZE	Random	Emb'd	LSIZE	Random	Emb'd	LSIZE
c7552	99.28	99.83	583	99.44	99.87	546	99.61	99.87	433
cs09234	90.68	98.55	1,097	95.30	99.26	824	96.55	99.39	683
cs13207	95.58	99.31	889	97.62	99.66	541	98.53	99.70	367
cs15850	96.29	99.36	1,107	98.34	99.67	783	98.81	99.70	686
cs38417	97.50	99.46	4,135	98.57	99.54	3,170	98.93	99.65	2,697
cs38584	93.01	98.74	894	95.10	99.43	878	96.47	99.67	590
\emptyset	95.39	99.21		97.40	99.57		98.15	99.66	

consideration, global fault efficacy G -FE from Equation (5.4.7) is given before (column “Random”) and after (column “Emb’d”) pattern embedding; furthermore in column “Emb’d LSIZE” the size of the synthesized bit-flipping logic is quoted in gate equivalents (GE). Note that the sequences before pattern embedding are obtained without BFL, consequently LSIZE is zero for this scenario. Average G -FE can be found in the last row of the table.

Interestingly, we observe that the G -FE is consistently larger than the single-stuck-at fault coverage for all circuits and all considered sequence lengths. When focusing on the results before pattern embedding, it appears that the distribution of random pattern resistant faults is different for single-stuck-at and resistive bridging faults. The lowest values of G -FE may always be observed for cs09234 and cs38584 disregarding the sequence length. Similarly, cs09234 consistently exhibits the lowest single-stuck-at fault coverage as well. Circuit cs38584, however, ranks second highest in single-stuck-at fault coverage. This suggests, that one has to be careful when transferring results on the magnitude of random pattern resistant faults from single-stuck-at to resistive bridging faults. Results obtained with our simulator SUPERB (see Chapter 7.2) demonstrate that for ISCAS 89 circuits, correlation is indeed limited while for the industrial circuits from NXP semiconductors a rather good correlation could be observed.

After pattern embedding, the global fault efficacy is significantly increased. Average values can also be found as a bar graph in Figure 10.2. Nevertheless, we may not only attribute improvements in fault coverage to the embedded deterministic patterns. Independent of the pattern embedding, the coverage of resistive bridging faults rises with increasing sequence length. The absolute impact of sequence length, however, is smaller when deterministic patterns are embedded. Without pattern embedding, average fault coverages increase from 95.39% for a 1,000 pattern to 98.15% for a 10,000 pattern sequence. With embedding, these numbers are 99.21% and 99.66%, respectively.

Similarly, the size of the synthesized bit-flipping logic is dependent on the sequence length: A longer sequence implies a smaller BFL (up to a factor of 2.4 for cs13207) – Figure

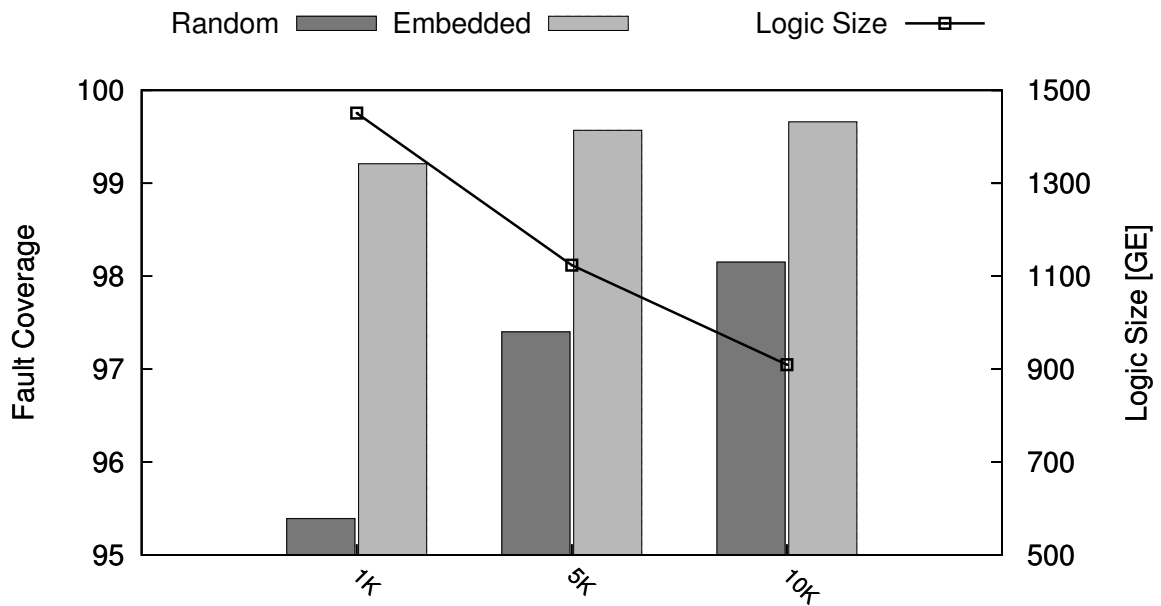


Figure 10.2: Average fault coverage G -FE and average logic size from Table 10.2.

10.2 illustrates the average logic size as a line graph. This may be explained by two observations:

1. A longer pseudo-random pattern sequence involves more degrees of freedom for the pattern embedding procedure.
2. Prior to pattern embedding, a larger sequence already implies higher single-stuck-at fault coverage. Consequently less faults have to be targeted during the BFL synthesis for longer sequences.

In summary, we observed that longer DLBIST sequence means less area for the DLBIST logic and enhanced coverage of non-target defects. The downside of this is, however, that longer test sequences also mean increased test application time.

10.2 Non-Target Defect Coverage Impact of X-Masking

The applicability of logic BIST to circuits containing random logic is substantially compromised by unknown values (also called unknowns). Sources of unknown values include tri-stated or floating buses, uninitialized flip-flops or latches, signals that cross clock domains in circuits with multiple clock domains, and X-values coming from analog or memory blocks that are embedded in the random logic circuit. If unknown values reach the test response evaluator (TRE), it may be no longer possible to distinguish between a fault-free and a faulty circuit. The most popular TRE, the MISR (see e.g. [11, 114] for an introduction), is particularly vulnerable to unknowns. Even a single unknown value may render the complete signature invalid.

There exist two general approaches which try to compensate for the detrimental effect of unknown values. On the one hand, there are *X-tolerant compactors*. These TREs are space compactors³ which can cope with a certain amount of unknowns occurring in addition to a number of faulty values. For instance X-COMPACT [122] and the Convolutional Compactor [140, 142] belong to this class of TREs. The silicon area requirements for X-tolerant compactors depend on the number of unknown values which can be tolerated. Typically their area overhead is higher than that of space compactors without X-tolerance.

On the other hand, unknown values may also be *masked out* by dedicated logic blocks [31, 126, 138, 191, 198]. This solution does not imply restrictions on the TRE used and thus can be applied to both space and time compaction.⁴ Masking is, however, test set specific. The silicon area requirements of this technique are very much dependent on the implementation, e.g. whether data is stored on the tester or on the chip. Furthermore, masking techniques may be deployed even if access to the circuit under test is restricted, e.g. due to intellectual property protection issues.

In the following, we will focus on a technique to mask unknown values, referred to as the *X-masking* technique. Yet, as will be discussed below, the problems investigated in our study do apply to X-tolerant compactors as well. X-masking is enabled by the *X-masking logic* (XML) which is a combinational block operated by the BIST control logic. The XML may selectively replace logic values supplied by the outputs of the circuit under test (CUT) by well-defined values, thus masking unknowns. In theory it is possible to exactly mask the unknown values produced by the CUT only. This, however, implies a large silicon area cost and is in general unnecessary to guarantee complete single-stuck-at fault coverage. The majority of faults is detected by several test vectors and at more than one circuit output. Therefore, we can afford to mask some of the known bits to obtain a compact XML without losing any single-stuck-at fault coverage (this is similar to [126, 138]). Unfortunately this approach might very well affect the coverage of non-target defects which is not considered during XML construction. This chapter presents a study we published in [W4], [P5] and [J2] on the impact of X-masking performed by the XML on the detection of non-target

³A space compactor takes a bit string from \mathbb{B}^n as input and compacts it into a bit string from \mathbb{B}^m , with $m < n$.

⁴Time compactors, like a MISR, compact long sequences of output responses into a single short signature.

defects.⁵ We use resistive bridging faults as surrogates for these defects. We explore the impact of masking known bits by computing the resistive bridging fault coverage for several XML configurations. In particular, we take into account silicon area requirements involved with each XML configuration. Note that during XML construction no information on the coverage of resistive bridging faults is considered.

This study is the first to evaluate the tradeoff between non-target defect coverage and logic size. Even though our results are generated for the masking technique, the issue of a potential decrease of non-target defect coverage also applies to X-tolerant compactors. These compactors connect each circuit output to multiple XOR trees. As a consequence, unknown values at outputs may invalidate detections at other circuit outputs connected to the same XOR gates. Providing more compactor outputs (XOR trees) will increase the circuit area and reduce the probability that a defect is missed. Hence, the trade-off between area cost and non-target defect coverage, which is investigated here for the case of X masking, exists also for X-tolerant compactors.

The contents of this chapter are structured as follows. The next Chapter 10.2.1 will introduce outline, and construction of the X-masking logic in more detail. Subsequently, setup and results of the resistive bridging fault simulation experiment are discussed in Chapter 10.2.2. Conclusions are found in Chapter 10.3.

10.2.1 The X-Masking Logic

We consider X-masking in combination with deterministic logic BIST (DLBIST) based on bit-flipping as introduced in Chapter 10.1. Yet, as X-masking does not impose any constraints on both pattern generator and test response evaluator, it can be adapted to any logic BIST or test compression architecture.

The structure of a circuit with DLBIST and XML is illustrated in Figure 10.3. LFSR, bit-flipping logic, XML and TRE are operated by the BIST control logic. This block is supported by the pattern counter, which stores the index of the current pattern, and the bit counter, which keeps track of the scan shift/capture cycles. The LFSR supplies pseudo-random patterns to the circuit's inputs which may have been manipulated by the bit-flipping logic in order to increase their fault coverage. Every output of the CUT is connected to an OR gate which in turn is linked to the TRE (e.g. a MISR). Each OR gate can be addressed individually by the XML to selectively mask the CUTs outputs.

Construction of X-Masking Logic

The XML is a combinational block which takes pattern (PC), and bit count (BC), as well as the state of the LFSR (LF) as inputs. Based on this information the XML derives the output bits to be masked. A value supplied by an output of the CUT is masked, iff the XML drives logical value 1 at the OR gate, which links the respective output to the MISR.

⁵We acknowledge the contribution of V. Gherman (University of Stuttgart, Germany) to this work.

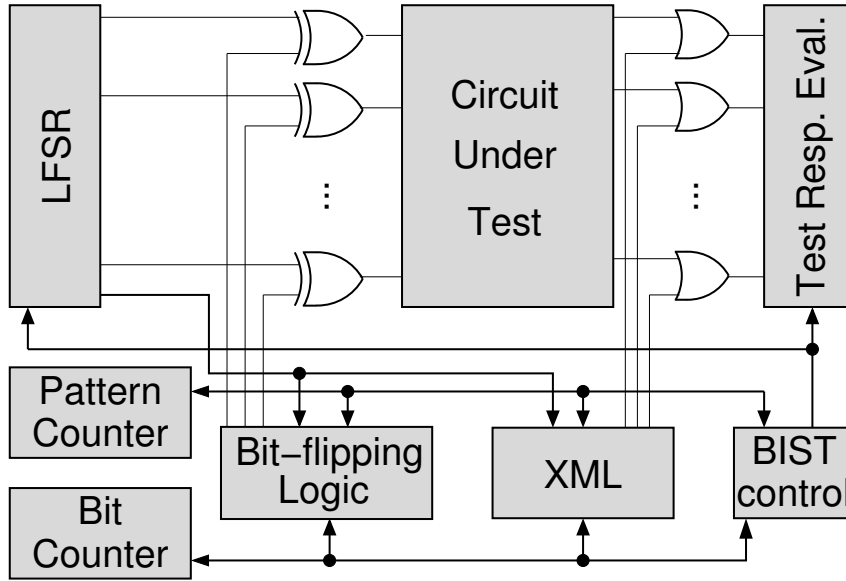


Figure 10.3: Deterministic logic BIST with X-masking logic.

Since the XML is not on the circuit's critical path the only extra delay (provided that the XML is not slower than the CUT) is introduced by the OR gates.

Let the CUT have o outputs and let the pattern set under consideration have p patterns. (Note that the term “output” stands for “primary output” for combinational and non-scan sequential circuits, scanout ports for full-scan circuits and primary outputs and scan-outs for partial-scan circuits.) Furthermore, let the responses of the circuit be $(r_{11}, r_{12}, \dots, r_{1o})$, $(r_{21}, r_{22}, \dots, r_{2o})$, \dots , $(r_{p1}, r_{p2}, \dots, r_{po})$, where $r_{ij} \in \{0, 1, X\}$ is the value that is driven at the j -th output of the fault-free CUT as a response to the i -th test pattern. Value X denotes the unknown value, i.e. either logical value 0 or 1. The XML computes a function $\text{XML} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{B}$ such that $\text{XML}(i, j) = 1$ if $r_{ij} = X$, which means that all unknown values must be masked. We refer to these r_{ij} as *masked bits*. Some of the r_{ij} supplied by the CUT have to be preserved in order to maintain the fault coverage. These so-called *relevant bits* must not be masked, i.e. $\text{XML}(i, j) = 0$ has to hold. All r_{ij} which neither belong to the set of masked bits, nor to that of the relevant bits, are subsumed by the *don't care* (DC) set. For each r_{ij} from this set the XML may drive either logical value 0 or 1.

The construction of the XML can be formulated as an instance of logic synthesis with don't cares [17]. Each r_{ij} , which we will denote a *bit* in the following, is uniquely identified by a combination of LFSR, pattern counter, and bit counter state, i.e. a value from the state-triple $LF \times PC \times BC$. The ON set consists of all state-triples for which $r_{ij} = X$, i.e. it corresponds to the set of masked bits. The OFF set is defined by the bits contained in the set of relevant bits. All other state-triples constitute the DC set.

Logic synthesis may be performed as soon as the ON and the OFF set are known. In general, smaller ON and OFF sets leave more degrees of freedom to the synthesis procedure and thus result in a more compact XML. While the ON set consisting of all masked bits is fixed, there are several possible collections of relevant bits which preserve the single-stuck-at fault

coverage. It is, however, not only the amount of relevant bits which should be minimized, but also the number of patterns from which relevant bits are selected. Therefore, a careful selection of relevant bits is crucial to ensure minimal silicon area cost.

Apart from the size of the XML, the impact of X-masking on the coverage of non-target defects is of key interest to us. Even though X-masking is guaranteed to preserve full single-stuck-at fault coverage, in general, it will reduce the number of times a single-stuck-at fault is detected. Multiple single-stuck-at fault detection, known as n -detection [106, 147], has been demonstrated to have a positive effect on the coverage of defects. Recent studies are reported in [8, 13] and data on the coverage of resistive bridging faults by n -detection test pattern can be found in Chapter 8.2.1. To limit the reduction in multiple detections, and thus preserve as much of the non-target defect coverage as possible, we introduce the following optimization criterion: We define a number $n \geq 1$ of detections that have to be retained when X-masking is performed. Assume, for instance, a given single-stuck-at fault is detected five times without X-masking and we set $n = 3$. Then X-masking may reduce the number of detections to four or three but not below. Increasing n leads to a higher number of single-stuck-at fault detections (and hence hopefully to a better coverage of non-target defects) but also to a larger set of relevant bits and thus to higher silicon area requirements for the XML.

The relevant bits are selected from the set of all $o \cdot p$ bits (excluding the masked bits). We employ two different strategies to identify the set of relevant bits (we published a detailed description of the strategies in [J2]):

Bit-based strategy: For a given n and a given single-stuck-at fault f , this strategy aims at selecting bits from as few patterns as possible. This is achieved by favoring patterns which have already been selected and/or detect many stuck-at faults. Furthermore, the strategy opts for patterns which contain low numbers of unknown values and detect f at an output at which only few unknown values may be observed for all patterns. This improves the decoupling of the ON and the OFF set.

Pattern-based strategy: For a given n and a given fault f , the method selects all bits from at least n patterns in which at least one bit detects the fault. If there are less than n such patterns then all the bits from all the patterns are selected. If the number of such patterns exceeds n , patterns are preferred which have already been selected, detect many faults, and have a low number of unknown values (which helps to decouple the ON and the OFF set).

While the bit-based strategy exhibits in general higher computational complexity, the pattern-based method classifies more bits as relevant for the same value of n .

In the following, we will explore the tradeoff between the value of n which controls both the amount of masked known bits, as well as the size of the XML, and the resistive bridging fault coverage. As additional parameters, we will consider the amount of unknown values and the aforementioned relevant bit selection strategies.

10.2.2 Experimental Results

We applied the XML synthesis technique to ISCAS 85 [19] circuits and the combinational cores of the ISCAS 89 [18] benchmarks. These circuit do not have tri-state buses or multiple clock domains and consequently do not produce unknown values at their outputs. Therefore, we assume a scenario in which a logic block producing X-values at its outputs feeds the inputs of the circuits under consideration. Experimental results discussed in Chapter 10.1 have indicated that when using DLBIST, the non-target defect coverage is very sensitive to the length of the LFSR generated pseudo-random sequence. To focus this study on the impact of X-masking, we have opted to use deterministic single-stuck-at test patterns only. They were generated by a commercial tool; afterwards X-values were injected randomly into the patterns. Subsequently, the resulting test sets (with injected unknown values) were simulated using a three-valued simulator (similar to the one from [167]) to obtain output responses containing X-values. This approach ensures realistic correlation of unknown values in the output responses targeted by XML synthesis. We conducted two X injection experiments:

- 1) For each pattern, X-values were randomly injected at 1% of the bit positions.
- 2) X-values were randomly injected at, on average, 3% of the bit positions. This was achieved as follows: For each pattern we randomly chose a value y from $[0, 6]$ (with uniform probability) and set $y\%$ of the bit positions to an X-value – which results in an average of 3% unknown values.

The XML circuitry has been synthesized using an approach based on binary decision diagrams [20, 39] (BDDs) which is discussed in [53] (some of the features described in that paper were not available when the experiments were performed). For selecting relevant bits, we employed both the bit-based and the pattern-based approach (explained in Chapter 10.2.1) for different values of n .

Setup of Resistive Bridging Fault Simulation Experiment

To evaluate the impact of the XML on the coverage of non-target defects, we simulated 10,000 randomly selected non-feedback resistive bridging faults (where available) utilizing our simulator [J3].⁶ Alternatively, we could also use layout extracted bridging faults (refer to Chapter 3.3). We employed the Shockley technology model (refer again to Chapter 5.2.2) and parameters from the SPICE model card of a $0.35\ \mu\text{m}$ technology from austriamicrosystems AG (AMS) to determine the critical resistances. Probability density function ρ proposed in [165] has been employed when calculating fault coverages. However, all experiments in this chapter could be repeated with any other short resistance distribution.

The resistive bridging fault model cannot handle unknown values in a meaningful way. This is because critical resistances can only be calculated if the driving strength of both driving gates is known. Therefore, the input assignment to these gates has to be completely

⁶Note that unless otherwise specified the fault lists used in this chapter are equal to those employed in all other experiments discussed in Part II of this work.

specified, even if a controlling value is applied to one of the gates' inputs. Therefore, we performed a Monte Carlo simulation [117] experiment. Given a test set IP we randomly assigned well-defined values from \mathbb{B} to all X-values, yielding a completely specified test set IP_1 . Subsequently, we conducted a resistive bridging fault simulation for IP_1 . Both steps were repeated $M - 1$ times such that in total M different, completely specified test sets IP_1, \dots, IP_M were simulated. Finally, resistive bridging fault coverage was obtained by averaging the fault coverages computed in each individual fault simulation run.

Algorithm 10.1: Monte Carlo estimation of non-target defect coverage.

Input: Input Pattern set IP which contains X-values;

Set X_{Base} of output bits with X-values;

For k XMLs, sets X_1, X_2, \dots, X_k of bits to be masked;

The number of repetitions M

Output: Average RBF coverage $FC_{\text{Base}}^\emptyset$ of the base scenario;

For k XMLs, average RBF coverages $FC_1^\emptyset, FC_2^\emptyset, \dots, FC_k^\emptyset$

```

1  $FC_{\text{Base}} := FC_1 := \dots := FC_k := 0;$ 
  /* Perform simulations for  $M$  instances of IP with X-values randomly
     assigned to 0s/1s */
2 for ( $i := 1; i \leq M; i := i + 1$ ) do
3   |  $IP_i := IP$  with X-values randomly assigned to 0s/1s;
4   |  $FC_{\text{Base}} := FC_{\text{Base}} + RBFSim(IP_i, X_{\text{Base}});$ 
5   | for ( $j := 1; j \leq k; j := j + 1$ ) do  $FC_j := FC_j + RBFSim(IP_i, X_j);$ 
6 end
  /* Compute average RBF coverages */
7  $FC_{\text{Base}}^\emptyset = FC_{\text{Base}}/M;$ 
8 for ( $j := 1; j \leq k; j := j + 1$ ) do  $FC_j^\emptyset := FC_j/M;$ 
9 return  $FC_{\text{Base}}^\emptyset, FC_1^\emptyset, FC_2^\emptyset, \dots, FC_k^\emptyset;$  /* Return average RBF coverages. */

```

In the absence of the XML circuitry X-values are observed at certain bits when the original test set IP is simulated. Detections of resistive bridging faults at these bits should not be accounted for when simulating test sets IP_1, \dots, IP_M . We refer to testing without XML as to the *base scenario*, and denote the output bits with unknown values as X_{Base} . Moreover, in the presence of the XML, detection of resistive bridging faults is impossible at all masked bit positions. This has to be respected during resistive bridging fault simulation. We synthesized several different XML architectures using both bit-based and pattern-based bit selection strategies, and different values of n . Let the number of these architectures be k , and let X_i be the set of bits masked by the i -th XML, $1 \leq i \leq k$. (Note that $X_{\text{Base}} \subseteq X_i$ always holds). To account for outputs which do not contribute to fault detection, we modified our simulator [J3] such that we may specify for each simulated pattern a set of outputs at which fault detections are discarded. Note that the same extension could be integrated into the simulator SUPERB introduced in Chapter 7.

For a given circuit the flow of the experiment may be described by Algorithm 10.1. It takes the original pattern set IP (including X-values), the bit positions X_{Base} at which IP

produces X-values, and the bits X_1, \dots, X_k masked by the considered XML configurations as input. Furthermore, the number of Monte Carlo simulations M has to be specified; for our experiments we used $M = 100$. Initially (see line 1) variables FC_{Base} , and FC_1, \dots, FC_k are initialized. They store the accumulated resistive bridging fault coverages for the base scenario and the k XML configurations, respectively. At the beginning of the outer loop (lines 2 – 6) a new test pattern instantiation IP_i is obtained from IP by randomly assigning unknown values (line 3). For each instantiation of IP we conduct $k + 1$ resistive bridging fault simulations using simulator [J3]. In the following, we denote our tool as $RBF\text{Sim}()$ to indicate that we modified it to discard detections at masked output bits. First the resistive bridging fault coverage of IP_i for the base scenario is computed. This is performed by executing $RBF\text{Sim}()$ using IP_i and X_{Base} as input (line 4), and accumulating the resulting coverage in FC_{Base} . Subsequently, $RBF\text{Sim}()$ is repeatedly executed for IP_i and each of the XML architectures X_j , $1 \leq j \leq k$, in line 5. The respective fault coverages are accumulated in FC_j . Note that FC_{Base} is always greater or equal than any FC_j . The difference $FC_{\text{Base}} - FC_j$ is an indicator for the coverage loss of non-target defects due to masking out specified (non-X) values by the j -th XML. Finally, in lines 7 and 8 the average fault coverage FC_{Base}^θ for the base scenario and the respective average fault coverage FC_j^θ for each of the k XML architectures is computed. These values are returned by the algorithm.

Results of Resistive Bridging Fault Simulation Experiment

Results for X injection experiment 1 (X-values randomly injected at 1% of the inputs) and the pattern-based relevant bit selection strategy can be found in Table 10.3. All experiments in this chapter have been performed for $M = 100$, i.e. for 100 instances of test set IP . In the first column of the table the name of the circuit can be found. The succeeding columns two and three quote the number “Bits” of bits at which X-values are observed in the base scenario and the average resistive bridging fault coverage “FC” for this scenario. We synthesized XML architectures for $n = 1$, $n = 3$, $n = 5$, and $n = 10$. For each architecture the table gives logic size “LS” of the synthesized XML circuitry in gate equivalents (GE), followed by the number of bits masked by the XML (“Bits”) and the average resistive bridging fault coverage “FC”. The last row of the table indicates average fault coverages. For all but three circuits (c3540, c6288, and c7552) we were able to compute G -ADI using RBF-ATPG discussed in Chapter 8 and thus calculated average global fault efficacy according to Equation (5.4.7). For the remaining three circuits excitation based fault coverage E -FC from Equation (5.4.3) had to be used instead. Recall that this implies that for these circuits the fault coverages quoted may be below their actual value. However, as the base scenario and all XML measurements are affected by this to the same extent, comparing the numbers is still meaningful.

Table 10.4 gives the results for experiment 1 and the bit-based relevant bit selection strategy. The structure of the table is similar to the one of Table 10.3, however, for this bit selection strategy we additionally synthesized XML architectures for $n = 10$, $n = 15$, and $n = 20$.

Independent of the relevant bit selection strategy, we can observe that with increasing n logic size grows. Yet, this growth is very moderate and much slower than n . Average

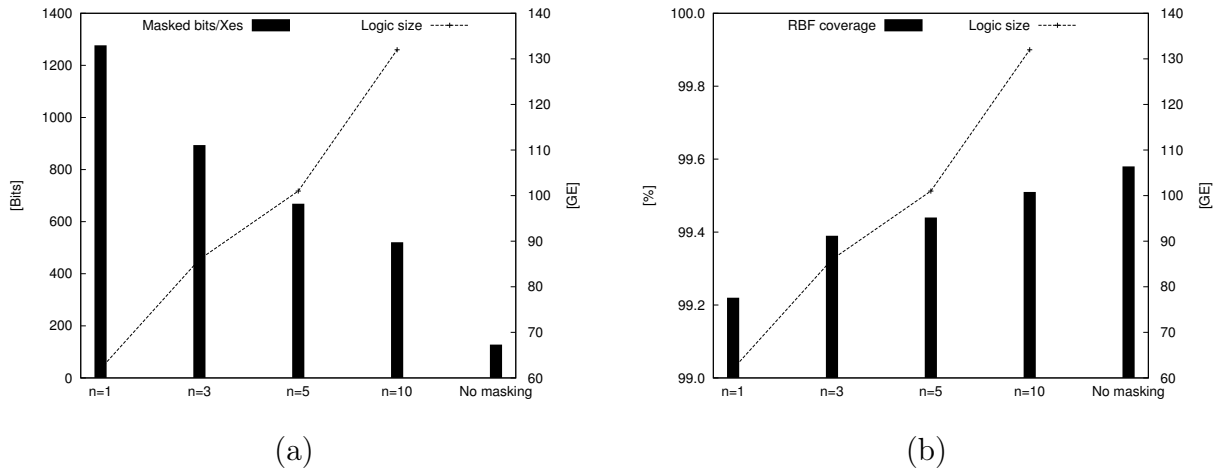


Figure 10.4: Results for c1355, pattern-based relevant bit selection and 1% Xs: Number of masked bits and logic size as function of n (a); RBF coverage and logic size as function of n (b).

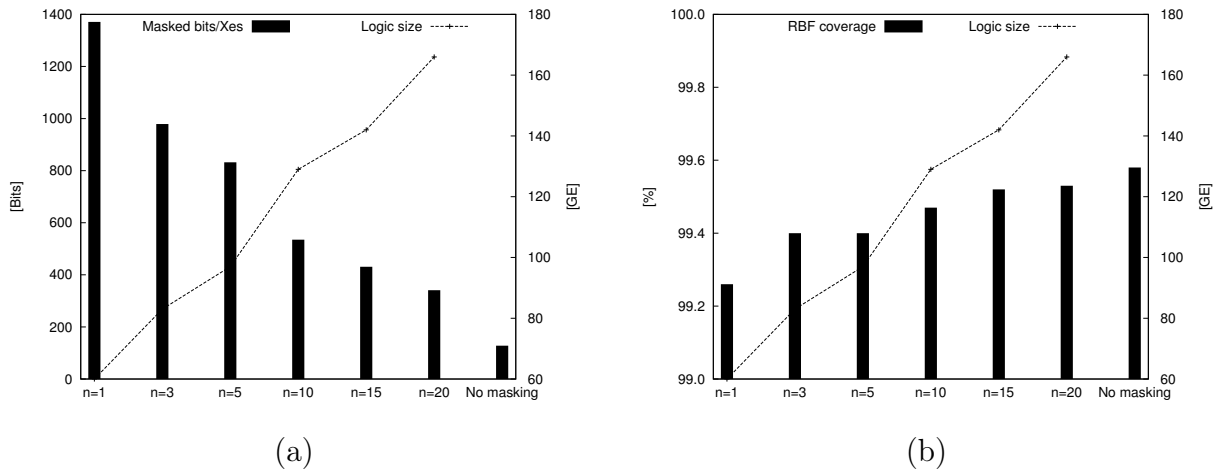


Figure 10.5: Results for c1355, bit-based relevant bit selection and 1% Xs: Number of masked bits and logic size as function of n (a); RBF coverage and logic size as function of n (b).

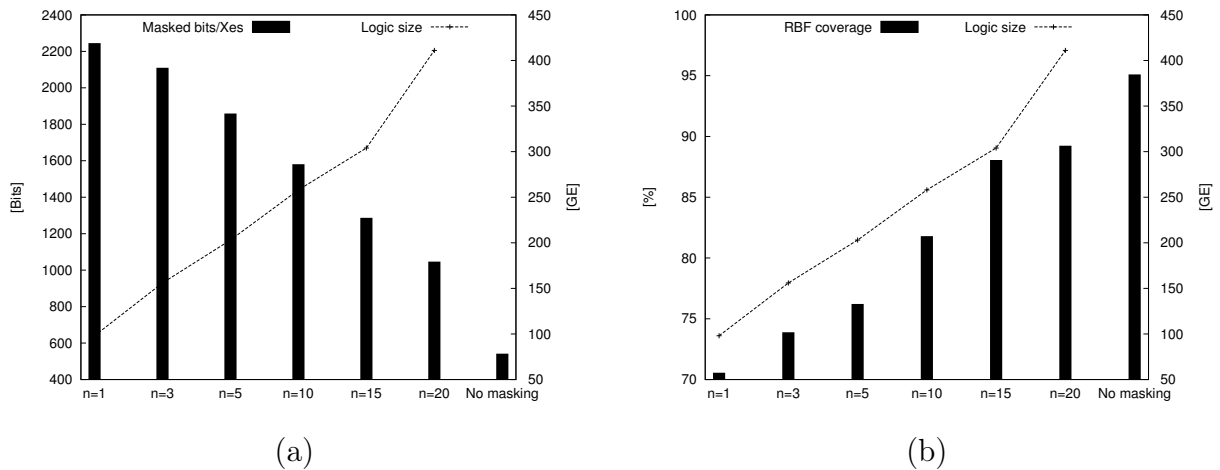


Figure 10.6: Results for c1355, bit-based relevant bit selection and 3% Xs: Number of masked bits and logic size as function of n (a); RBF coverage and logic size as function of n (b).

increase in logic size is higher for the bit-based strategy: for $n = 10$ the XML circuit is on average 2.27 times larger than for $n = 1$. In contrast to that, the average logic size for the pattern-based strategy exactly doubles when going from $n = 1$ to $n = 10$. In terms of the absolute size the XML circuitry for the pattern-based strategy is (depending on n) 5% to 19% larger than that created for the bit-based strategy. Resistive bridging fault coverage is hardly affected by the XML. For both relevant bit selection strategies average coverage slightly drops for $n = 1$ and almost recovers for $n = 3$. Similar observations can be made for the bit-based strategy. For c1355, the tradeoff between size of the XML circuitry, number of masked bits and fault coverage is illustrated in Figures 10.4 (for the pattern-based strategy) and 10.5 (for the bit-based strategy).

Table 10.5 reports experimental results for X injection experiment 2 (X-values randomly injected at an average of 3% of the inputs) and the bit-based relevant bit selection strategy (in this experiment no results for pattern-based strategy were generated). The structure of the table is identical to that of Table 10.4. In contrast to experiment 1, the coverage drop for $n = 1$ is large. On average 4.97 percentage points are lost due to the introduction of the XML. High values of n are required to compensate for this loss in non-target defect coverage. The growth in logic size is larger than in experiment 1, however, still considerably lower than the increase of n . Again, results for c1355 are illustrated in graph form – see Figure 10.6.

In summary the results suggest, that the loss in resistive bridging fault coverage due to X-masking is very moderate if low fractions of unknown values are to be expected. This is true, even if n -detection is not considered ($n = 1$). In combination with n -detection, however, low values of n are sufficient to (nearly) compensate for the coverage loss due to the XML. By contrast, higher percentages of X-values necessitate the preservation of multiple detections. Otherwise the coverage of non-target defects may be compromised.

10.3 Conclusions

We evaluated two BIST techniques in terms of their effect on non-target defect detection. Resistive bridging faults were used as a surrogate for non-target defects. Both BIST techniques guarantee maximum single-stuck-at fault coverage, but are not optimized with respect to defect detection.

In the first part of the chapter we investigated the non-target defect coverage of test vectors supplied by deterministic logic BIST using bit-flipping. This technique maximizes the single-stuck-at fault coverage of a sequence of pseudo-random patterns generated on-chip by embedding deterministic single-stuck-at test patterns into the sequence. Experimental results show that both the random pattern sequence as well as the embedded single-stuck-at patterns are required to attain sufficiently high levels of resistive bridging fault coverage. Coverage can be enhanced additionally if longer sequences of pseudo-random patterns are used. This suggests that deterministic logic BIST is superior to conventional single-stuck-at test application. Based on our findings, we advocate to exploit the available test time to the extent possible to maximize the sequence length and thus increase defect coverage. Additionally, this also helps to reduce the silicon area cost for the required bit-flipping logic.

Subsequently, the second part of the chapter dealt with an X-masking technique based on the X-masking logic (XML). X-masking prevents unknown logical values from entering the test response evaluator replacing them by well-defined logical values. This enables the use of arbitrary response evaluators including those vulnerable to unknown values. To reduce the silicon area required to implement the XML, some known bits are allowed to be masked in addition to the unknown bits as long as the single-stuck-at fault coverage is not compromised. We considered a X-masking technique which modulates the amount of masked known bits, and thus the area overhead, using a metric based on n -detection. For several values of n we explored to what extent the masking of known logical values degrades the coverage of non-target defects. Experimental results demonstrate that relatively small values of n are sufficient to compensate for the reduction in non-target coverage imposed by the X-masking logic – as long as the fraction of unknown values to be masked is low. For a higher percentage of unknowns, n has to be increased substantially to counterbalance the impact of the XML. This, however, is paid by increased silicon area demands.

In summary, our experiments demonstrated that the BIST techniques under consideration hardly degrade and even improve non-target defect coverage. Yet, we also found that this is only true if sequence length and value of n are carefully selected. This suggests that the impact of a BIST technology on the non-target defect coverage has to be evaluated using e.g. the resistive bridging fault model to avoid compromising defect detection.

Table 10.3: Experimental results, pattern-based relevant bit selection (1% X values at the inputs).

Circuit	Base			$n = 1$			$n = 3$			$n = 5$			$n = 10$		
	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	
c0432	28	95.72	23	55	95.69	26	31	95.72	28	30	95.72	30	28	95.72	
c0499	63	99.29	37	556	99.29	53	361	99.29	62	310	99.29	75	176	99.29	
c0880	20	96.63	25	88	96.37	29	39	96.61	33	30	96.62	35	23	96.63	
c1355	128	99.58	62	1277	99.22	86	894	99.39	101	669	99.44	132	521	99.51	
c1908	183	99.44	122	881	99.41	168	656	99.42	187	547	99.44	219	394	99.44	
c2670	160	97.89	128	2021	97.73	181	1103	97.88	199	746	97.89	243	420	97.89	
c3540	205	96.94	157	588	96.91	171	383	96.94	205	322	96.94	212	278	96.94	
c5315	406	99.27	228	2657	98.92	365	1485	99.13	428	1107	99.19	494	770	99.27	
c6288	143	90.38	51	188	90.25	53	151	90.38	56	143	90.38	56	143	90.38	
c7552	602	98.81	358	3561	98.66	468	2056	98.79	528	1633	98.80	582	1100	98.81	
cs00298	8	97.48	10	37	97.45	11	18	97.48	11	12	97.48	12	11	97.48	
cs00344	11	95.68	13	37	94.60	14	18	95.66	14	14	95.66	15	11	95.68	
cs00400	17	98.28	26	75	98.19	27	43	98.27	28	37	98.24	30	29	98.28	
cs00444	9	97.82	13	66	97.76	16	35	97.82	18	34	97.82	20	17	97.82	
cs00526	23	98.35	35	180	98.29	45	96	98.34	48	77	98.35	49	54	98.35	
cs00713	15	98.68	20	135	98.57	27	69	98.67	29	48	98.67	31	25	98.68	
cs05378	2024	98.97	455	8292	98.84	608	5262	98.94	716	4410	98.95	863	3320	98.96	
cs13207	2808	99.10	1648	102315	99.04	2364	61531	99.08	2796	46632	99.09	3461	26858	99.10	
cs15850	2115	98.74	1540	32681	98.58	2059	18422	98.69	2328	13832	98.72	2895	8363	98.73	
cs38584	5626	96.47	3746	84430	96.14	5535	46974	96.38	6524	33201	96.43	7917	20763	96.45	
\emptyset		97.68		97.50	97.64			97.64			97.66			97.67	

Table 10.4: Experimental results, bit-based relevant bit selection (1% X values at the inputs).

Circuit	Base		$n = 1$		$n = 3$		$n = 5$		$n = 10$		$n = 15$		$n = 20$	
	Bits	FC	LS	Bits FC	LS	Bits FC	LS	Bits FC	LS	Bits FC	LS	Bits FC	LS	Bits FC
c0432	28	95.72	21	66 95.58	24	50 95.68	25	41 95.71	26	31 95.72	28	29 95.72	28	29 95.72
c0499	63	99.29	35	586 99.29	51	347 99.29	58	293 99.29	73	165 99.29	85	117 99.29	99	84 99.29
c0880	20	96.63	23	110 96.35	30	36 96.62	32	30 96.62	34	23 96.63	35	23 96.63	36	21 96.63
c1355	128	99.58	60	1371 99.26	83	979 99.40	97	832 99.40	129	535 99.47	142	431 99.52	166	341 99.53
c1908	183	99.44	110	1215 99.26	139	920 99.40	157	646 99.43	197	531 99.44	225	440 99.44	232	344 99.44
c2670	160	97.89	112	2690 97.67	156	1360 97.85	185	1021 97.88	226	682 97.89	247	490 97.89	262	344 97.89
c3540	205	96.94	118	1080 96.64	149	580 96.92	169	487 96.92	191	403 96.93	214	287 96.94	225	269 96.94
c5315	406	99.27	206	3430 98.69	330	2021 99.15	369	1502 99.10	451	1103 99.24	467	1096 99.23	496	979 99.20
c6288	143	90.38	49	211 90.23	52	159 90.36	54	144 90.38	56	143 90.38	56	143 90.38	56	143 90.38
c7552	602	98.81	336	4359 98.52	437	3044 98.77	493	2225 98.80	550	1713 98.80	595	1283 98.80	610	1040 98.80
cs00298	8	97.48	9	36 97.45	11	13 97.48	11	15 97.48	12	10 97.48	13	8 97.48	13	8 97.48
cs00344	11	95.68	12	65 95.34	13	22 95.64	14	20 95.68	15	11 95.68	15	11 95.68	15	11 95.68
cs00400	17	98.28	25	117 97.91	27	52 98.27	27	45 98.27	28	29 98.28	32	22 98.28	33	17 98.28
cs00444	9	97.82	13	97 97.71	15	50 97.82	16	40 97.82	20	16 97.82	21	14 97.82	22	10 97.82
cs00526	23	98.35	33	281 98.09	41	166 98.32	44	97 98.34	48	58 98.35	50	43 98.35	51	37 98.35
cs00713	15	98.68	18	154 98.58	24	84 98.66	27	60 98.68	31	24 98.68	31	23 98.68	32	16 98.68
cs05378	2024	98.97	338	10383 98.63	470	7810 98.87	555	6556 98.91	703	5170 98.96	801	4448 98.97	883	4032 98.97
cs13207	2808	99.10	1351	113718 98.96	2082	70397 99.07	2604	56385 99.09	3333	35440 99.10	3911	23522 99.10	4291	17866 99.09
cs15850	2115	98.74	1164	40005 98.44	1807	22553 98.67	2161	15609 98.68	2775	9264 98.71	3120	7181 98.73	3275	6014 98.73
cs38584	5626	96.47	3286	97955 95.42	5145	53161 96.33	6177	37775 96.41	7724	22766 96.43	8490	16409 96.45	9050	13590 96.45
\emptyset		97.68		97.40		97.63		97.64		97.66		97.67		97.67

Table 10.5: Experimental results, bit-based relevant bit selection (3% X values at the inputs).

Circuit	Base		$n = 1$		$n = 3$		$n = 5$		$n = 10$		$n = 15$		$n = 20$							
	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC	LS	Bits	FC						
c0432	61	94.01	33	122	93.20	41	88	93.86	43	78	94.00	45	70	94.01	46	66	94.01	47	63	94.01
c0499	314	94.53	75	1222	43.27	111	1023	85.57	123	912	88.20	183	703	88.59	231	525	94.10	275	405	94.52
c0880	63	96.34	51	240	95.63	64	110	96.25	67	117	96.28	81	81	96.34	83	71	96.34	83	66	96.34
c1355	542	95.10	98	2245	70.56	156	2110	73.89	203	1859	76.22	258	1581	81.80	304	1287	88.06	411	1047	89.24
c1908	360	99.10	136	1757	98.61	208	1261	99.01	249	1132	99.08	300	888	99.10	333	776	99.10	356	674	99.10
c2670	456	93.98	207	4477	92.24	286	3084	93.80	358	2176	93.92	440	1453	93.97	507	1205	93.98	541	774	93.98
c3540	433	96.18	219	1503	95.06	273	1120	95.80	314	930	96.08	381	735	96.15	414	640	96.17	419	564	96.17
c5315	925	98.81	305	5090	96.74	502	3820	98.14	623	3153	98.22	762	2238	98.51	846	1976	98.67	875	1678	98.73
c6288	326	88.25	72	427	85.74	77	344	88.12	82	333	88.25	89	327	88.25	89	326	88.25	89	326	88.25
c7552	1574	97.57	532	6801	95.14	792	5125	97.28	880	4866	97.45	1004	3984	97.50	1071	3325	97.53	1152	2409	97.53
cs00298	29	96.94	29	133	96.35	33	76	96.90	38	45	96.93	43	39	96.94	44	33	96.94	46	30	96.94
cs00344	20	95.69	21	71	94.75	26	33	95.67	29	27	95.69	30	20	95.69	30	20	95.69	30	20	95.69
cs00400	49	97.58	38	253	95.97	53	155	97.36	57	123	97.46	65	60	97.58	70	54	97.58	74	50	97.58
cs00444	41	97.01	36	191	95.60	43	89	96.88	48	86	96.87	53	49	97.00	54	45	97.01	54	42	97.01
cs00526	94	98.19	77	493	97.72	99	300	98.08	104	235	98.14	109	218	98.17	132	146	98.19	134	129	98.19
cs00713	87	98.31	57	447	97.74	84	273	98.19	99	174	98.30	105	126	98.31	107	123	98.31	111	99	98.31
cs05378	2926	98.54	518	16112	97.64	792	13659	98.35	997	11497	98.38	1299	8700	98.50	1484	7034	98.53	1658	6331	98.53
cs13207	8536	98.93	2041	166074	97.76	3751	131954	98.20	4874	113199	98.25	6988	81267	98.29	8397	55940	98.30	9531	43721	98.30
cs15850	5872	98.38	1786	64684	96.22	3135	46191	97.34	4056	36472	97.57	5539	23249	97.74	6527	17229	97.74	7113	14826	97.75
cs38584	12525	95.95	5053	149702	94.14	8766	96010	95.69	11078	74158	95.82	14586	45301	95.92	16423	32520	95.92	17464	27853	95.90
\emptyset		96.47			91.50			94.72			95.06			95.42			96.02			96.10

11 Summary and Discussion of Part II

While Part I of this thesis gave an overview over several bridging fault models, and particularly introduced the theoretical aspects of the resistive bridging fault model, the second part focused on the translation of this theory into the practical application domain. The applications considered can be roughly attributed to three areas:

1. The “traditional” testing disciplines of fault simulation and test pattern generation.
2. The integration of advanced testing concepts into the framework of the model.
3. The evaluation of testing methodologies with respect to their resistive short detection.

Our fault simulator SUPERB, and our test pattern generator RBF-ATPG introduced in Chapters 7 and 8, respectively, belong to the first area. Both tools are core elements of a testing methodology for resistive bridging faults without which all other applications of the model would not be thinkable. They could only be realized by developing efficient algorithms which exactly capture the parametric nature of the resistive bridging fault model, its sophisticated electrical concept, and the probabilistic fault coverage metrics. In particular, SUPERB demonstrates that fast and accurate resistive bridging fault simulation is feasible, and actually renders handling of multi-million gate designs possible. Due to the combination of the sectioning technique and parallel-pattern evaluation realized in SUPERB, we are now able to simulate resistive shorts in circuits of practically relevant size which enables testing of resistive shorts in practice.

The sectioning technique also forms the basis of the second component in this area: the test pattern generator RBF-ATPG. Further ingredients of this tool include a solver for Boolean satisfiability problems and an interval-based fault simulator. Test pattern generation is a key task and – as our experimental comparison with n -detection and 4-way test sets underlined – absolutely necessary to ensure complete coverage of resistive shorts. Besides that, it proved to be extremely valuable that RBF-ATPG accurately determines G -ADI for circuits which may not be exhaustively simulated. As a consequence, we can now compute the exact fault coverage metric G -FC for a wide range of circuits. This is a prerequisite to accurately grade test patterns, which is for instance required when evaluating testing methodologies.

In the second area we explored two extensions to the resistive bridging fault model. There it turned out to be beneficial that the model’s electrical concept is very powerful, yet flexible enough to permit the integration of additional physical effects. Because of the extensions proposed in Chapter 9 it is now possible to capture the influence of varying operating temperature and power supply voltage on the coverage of resistive shorts. Furthermore, we could also transfer the model’s basic electrical framework to a testing approach very

different from voltage testing: quiescent current (Delta-IDDQ) testing. This technique, however, is not an isolated part of the model. Rather, voltage and quiescent current testing are tightly integrated thanks to the parametric nature of the resistive bridging fault model. Thus, their effectivity can be directly compared. Furthermore, due to the comparability of detection intervals obtained in the voltage and the current testing domain, even sequential circuits can be handled accurately.

We also exploited this tight integration for the experiments which may be attributed to the third application area. Results reported in Chapter 9.2 underline that the combination of voltage and Delta-IDDQ testing is very effective in the detection of resistive shorts detectable by conventional voltage testing only (hard defects). Additionally, we were able to quantify the coverage of flaws by Delta-IDDQ testing. Flaws are defects which are undetectable by conventional voltage testing, yet pose a substantial reliability risk as they may lead to catastrophic failures during the future use of the device. In particular, experimental results demonstrate that voltage and current testing are orthogonal in the sense that for the same set of test patterns, they cover different resistive shorts. Therefore, even adding just a few Delta-IDDQ measurements to a voltage testing scenario can increase the overall coverage of resistive shorts.

In a scenario, however, which purely relies on voltage testing, flaws may also be covered by testing at several operating temperatures and/or power supply voltages. In the extended analytical framework of the resistive bridging fault model, these parameters can easily be modified. This facilitates experiments like the ones discussed in Chapter 9.1 which considered the impact of these advanced testing methodologies on the coverage of hard defects and flaws. Additionally, our experiments accounted for the costs involved with either technique. Extensive experimental data helps to find the optimal testing strategy in terms of coverage and costs for both products with elevated reliability requirements as well as high-volume manufacturing test.

The concept of the resistive bridging fault model even turns out to be beneficial in the context of testing methodologies which are optimized to guarantee maximal single-stuck-at fault detection, yet do not account for the coverage of defects. We demonstrated that the coverage of resistive shorts can be used to quantify the effect of two BIST techniques on the non-target defect detection. We used resistive shorts modeled by resistive bridging faults as a surrogate for these defects. Experiments reported in Chapters 10.1 and 10.2 explored for each technique several configurations with respect to their silicon area demands and the resulting non-target defect coverage. It turned out that the coverage is heavily dependent on the configuration and thus should be evaluated explicitly to ensure optimal defect detection along with low area demands.

In summary, the flexible and extensible concept of the resistive bridging fault model in combination with efficient algorithms yields an accurate analytical tool and enables targeted detection of resistive shorts.

Concluding Remarks

Taken together, this doctoral thesis demonstrates that the concept of the resistive bridging fault model is extremely flexible and powerful. At the same time, the model accurately reflects the impact of resistive shorts on the behavior of digital circuits. Thanks to the model's flexibility, we were able to extend its modeling capabilities to several advanced testing concepts. These novel extensions allow us to accurately assess the influence of variable operating conditions on the coverage of resistive shorts. Additionally, it is now possible to directly compare the effectivity of quiescent current and voltage testing. We developed several metrics which precisely evaluate the individual benefit of these advanced testing concepts side by side. In combination, our contributions turn the resistive bridging fault model into a highly accurate and extremely versatile analytical instrument. Moreover, the model's accuracy paired with its moderate complexity enables targeted detection of resistive shorts in practical applications.

The practicality of the resistive bridging fault model is guaranteed by our fault simulator SUPERB and our automatic test pattern generator RBF-ATPG. Powered by the efficient algorithms we developed in this thesis, these tools exactly replicate the parametric nature of the fault model. In particular, SUPERB demonstrates that fast and accurate resistive bridging fault simulation is feasible and actually renders handling of multi-million gate designs possible. The test patterns generated by RBF-ATPG completely cover all detectable resistive shorts. Furthermore, the outputs of RBF-ATPG are a prerequisite for the accuracy of our fault coverage metrics. Numerous experiments presented in this thesis underline the applicability of the resistive bridging fault model, while demonstrating the versatility of the model in various application domains. Our results help finding the optimal testing strategy when resistive shorts are targeted. Yet, they also allow us to improve the defect detection capabilities of techniques which are specializing in conventional stuck-at faults.

Despite our contributions, there are still some questions pending. First of all, we would like to explore the impact of process variations on the resistive bridging fault model. Furthermore, we plan to extend its modeling capabilities to the dynamic effects caused by resistive shorts (in a way similar to Li et al. [102]). We emphasize the need for a tight integration of dynamic effects into the parametric model – analogical to our approach for resistive open defects in [P19]. Moreover, we intend to continue our research on the benefits of the resistive bridging fault model for diagnosis. Locating shorts present in a circuit is a key task, yet our studies on this topic are still in the early stages (see our publications [W12, W13]). Finally, validation on manufactured silicon would provide the ultimate proof of the efficiency of the methods studied in this thesis.

Author's Publications

Items marked with (*) are not discussed in this thesis.

Journal Articles

- [J5] P. Engelke, I. Polian, M. Renovell, S. Kundu, B. Seshadri, and B. Becker. On detection of resistive bridging defects by low-temperature and low-voltage testing. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 27(2):327–338, Feb. 2008. doi: 10.1109/TCAD.2007.913382.
- [J4] P. Engelke, I. Polian, M. Renovell, and B. Becker. Automatic test pattern generation for resistive bridging faults. *Journal of Electronic Testing: Theory and Applications*, 22(1):61–69, Feb. 2006. doi: 10.1007/s10836-006-6392-x.
- [J3] P. Engelke, I. Polian, M. Renovell, and B. Becker. Simulating resistive bridging and stuck-at faults. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 25(10):2181–2192, Oct. 2006. doi: 10.1109/TCAD.2006.871626.
- [J2] Y. Tang, H.-J. Wunderlich, P. Engelke, I. Polian, B. Becker, J. Schlöffel, F. Hapke, and M. Wittke. X-masking during logic BIST and its impact on defect coverage. *IEEE Transactions on VLSI Systems*, 14(2):193–202, Feb. 2006. doi: 10.1109/TVLSI.2005.863742.
- [J1] I. Polian, P. Engelke, M. Renovell, and B. Becker. Modeling feedback bridging faults with non-zero resistance. *Journal of Electronic Testing: Theory and Applications*, 21(1):57–69, Feb. 2005. doi: 10.1007/s10836-005-5287-6.

Papers in Formal Proceedings (Refereed)

- [P20] A. Czutro, I. Polian, M. Lewis, P. Engelke, S. Reddy, and B. Becker. TIGUAN: Thread-parallel integrated test pattern generator utilizing satisfiability analysis. In *International Conference on VLSI Design*, pages 227–232, 2009. doi: 10.1109/VLSI.Design.2009.20. (*)
- [P19] A. Czutro, N. Houarche, P. Engelke, I. Polian, M. Comte, M. Renovell, and B. Becker. A simulator of small-delay faults caused by resistive-open defects. In *IEEE European Test Symposium*, pages 113–118, 2008. doi: 10.1109/ETS.2008.19. (*)

- [P18] P. Engelke, I. Polian, J. Schlöffel, and B. Becker. Resistive bridging fault simulation of industrial circuits. In *Design, Automation and Test in Europe*, pages 628–633, 2008. doi: 10.1109/DATE.2008.4484747.
- [P17] S. Hillebrecht, I. Polian, P. Engelke, B. Becker, M. Keim, and W.-T. Cheng. Extraction, simulation and test generation for interconnect open defects based on enhanced aggressor-victim model. In *IEEE International Test Conference*, 2008. doi: 10.1109/TEST.2008.4700642. (*).
- [P16] S. Spinner, I. Polian, P. Engelke, B. Becker, M. Keim, and W.-T. Cheng. Automatic test pattern generation for interconnect open defects. In *IEEE VLSI Test Symposium*, pages 181–186, 2008. doi: 10.1109/VTS.2008.30. (*).
- [P15] P. Engelke, B. Braitling, I. Polian, M. Renovell, and B. Becker. SUPERB: Simulator utilizing parallel evaluation of resistive bridges. In *IEEE Asian Test Symposium*, pages 433–438, 2007. doi: 10.1109/ATS.2007.71.
- [P14] S. Spinner, J. Jiang, I. Polian, P. Engelke, and B. Becker. Simulating open-via defects. In *IEEE Asian Test Symposium*, pages 265–270, 2007. doi: 10.1109/ATS.2007.72. (*).
- [P13] P. Engelke, I. Polian, H. Manhaeve, M. Renovell, and B. Becker. Delta-IDDQ testing of resistive short defects. In *IEEE Asian Test Symposium*, pages 63–68, 2006. doi: 10.1109/ATS.2006.260994.
- [P12] M. Renovell, M. Comte, I. Polian, P. Engelke, and B. Becker. Analyzing the memory effect of resistive open in CMOS random logic. In *International Conference on Design and Test of Integrated Systems in Nanoscale Technology*, pages 251–256, Sept. 2006. doi: 10.1109/DTIS.2006.1708691. (*).
- [P11] M. Renovell, M. Comte, I. Polian, P. Engelke, and B. Becker. A specific ATPG technique for resistive open with sequence recursive dependency. In *IEEE Asian Test Symposium*, pages 273–278, 2006. doi: 10.1109/ATS.2006.261031. (*).
- [P10] G. Chen, S. Reddy, I. Pomeranz, J. Rajski, P. Engelke, and B. Becker. An unified fault model and test generation procedure for interconnect opens and bridges. In *IEEE European Test Symposium*, pages 22–27, 2005. doi: 10.1109/ETS.2005.6.
- [P9] S. Kundu, P. Engelke, I. Polian, and B. Becker. On detection of resistive bridging defects by low-temperature and low-voltage testing. In *IEEE Asian Test Symposium*, pages 266–269, 2005. doi: 10.1109/ATS.2005.83.
- [P8] I. Polian, S. Kundu, J.-M. Gallière, P. Engelke, M. Renovell, and B. Becker. Resistive bridge fault model evolution from conventional to ultra deep submicron technologies. In *IEEE VLSI Test Symposium*, pages 343–348, 2005. doi: 10.1109/VTS.2005.72.
- [P7] P. Engelke, I. Polian, M. Renovell, and B. Becker. Automatic test pattern generation for resistive bridging faults. In *IEEE European Test Symposium*, pages 160–165, 2004. doi: 10.1109/ETSYM.2004.1347652.
- [P6] P. Engelke, I. Polian, M. Renovell, B. Seshadri, and B. Becker. The pros and cons of very-low-voltage testing: An analysis based on resistive short defects. In *IEEE VLSI Test Symposium*, pages 171–178, 2004. doi: 10.1109/VTEST.2004.1299240.

- [P5] Y. Tang, H.-J. Wunderlich, H. Vranken, F. Hapke, M. Wittke, P. Engelke, I. Polian, and B. Becker. X-masking during logic BIST and its impact on defect coverage. In *IEEE International Test Conference*, pages 442–451, 2004. doi: 10.1109/TEST.2004.1386980.
- [P4] P. Engelke, I. Polian, M. Renovell, and B. Becker. Simulating resistive bridging and stuck-at faults. In *IEEE International Test Conference*, pages 1051–1059, 2003. doi: 10.1109/TEST.2003.1271093.
- [P3] I. Polian, P. Engelke, M. Renovell, and B. Becker. Modelling feedback bridging faults with non-zero resistance. In *IEEE European Test Workshop*, pages 91–96, 2003. doi: 10.1109/ETW.2003.1231674.
- [P2] I. Polian, P. Engelke, and B. Becker. Efficient bridging fault simulation of sequential circuits based on multi-valued logics. In *IEEE International Symposium on Multiple-Valued Logic*, pages 216–222, 2002. doi: 10.1109/ISMVL.2002.1011092. (*)
- [P1] P. Engelke, B. Becker, and M. Keim. A parameterizable fault simulator for bridging faults. In *IEEE European Test Workshop*, pages 63–68, 2000. doi: 10.1109/ETW.2000.873780. (*)

Workshop Contributions (Refereed)

- [W15] A. Czutro, I. Polian, M. Lewis, P. Engelke, S. Reddy, and B. Becker. TIGUAN: Thread-parallel integrated test pattern generator utilizing satisfiability analysis. In *edaWorkshop*, pages 69–74, 2008. (*)
- [W14] P. Engelke, I. Polian, J. Schlöffel, and B. Becker. Resistive bridging fault simulation of industrial circuits. In *GI/ITG Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen"*, pages 143–148, 2008.
- [W13] I. Polian, Y. Nakamura, P. Engelke, S. Spinner, K. Miyase, S. Kajihara, B. Becker, and X. Wen. Diagnosis of realistic defects based on the X-fault model. In *IEEE International Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pages 263–266, 2008. doi: 10.1109/DDECS.2008.4538798. (*)
- [W12] I. Polian, Y. Nakamura, P. Engelke, S. Hillebrecht, K. Miyase, S. Kajihara, B. Becker, and X. Wen. Diagnose realistischer Defekte mit Hilfe des X-Fehlermodells. In *GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf*, pages 155–156, 2008. (*)
- [W11] S. Spinner, I. Polian, P. Engelke, B. Becker, M. Keim, and W.-T. Cheng. Automatic test pattern generation for interconnect open defects. In *GI/ITG Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen"*, pages 47–52, 2008. (*)
- [W10] P. Engelke, I. Polian, H. Manhaeve, M. Renovell, and B. Becker. Delta-IDDQ testing of resistive short defects. In *IEEE International Workshop On Current and Defect-Based Testing*, 2006.

- [W9] P. Engelke, I. Polian, H. Manhaeve, M. Renovell, and B. Becker. IDDQ testing of resistive bridging defects. In *GI/ITG Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen"*, pages 123–124, 2006. (*).
- [W8] P. Engelke, V. Gherman, I. Polian, Y. Tang, H.-J. Wunderlich, and B. Becker. Sequence length, area cost and non-target defect coverage tradeoffs in deterministic logic BIST. In *IEEE International Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pages 11–18, 2005.
- [W7] P. Engelke, V. Gherman, I. Polian, Y. Tang, H.-J. Wunderlich, and B. Becker. Sequence length, area cost and non-target defect coverage tradeoffs in deterministic logic BIST. In *IEEE International Workshop on Current and Defect-Based Testing*, pages 43–48, 2005.
- [W6] P. Engelke, I. Polian, M. Renovell, and B. Becker. Automatic test pattern generation for resistive bridging faults. In *IEEE International Workshop On Current and Defect-Based Testing*, pages 89–94, 2004.
- [W5] P. Engelke, I. Polian, M. Renovell, B. Seshadri, and B. Becker. The pros and cons of very-low-voltage testing: An analytical view. In *GI/ITG Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen"*, pages 149–153, 2004.
- [W4] Y. Tang, H.-J. Wunderlich, H. Vranken, F. Hapke, M. Wittke, P. Engelke, I. Polian, and B. Becker. X-masking during logic BIST and its impact on defect coverage. In *IEEE International Workshop on Test Resource Partitioning*, pages 442–451, 2004.
- [W3] P. Engelke, I. Polian, M. Renovell, and B. Becker. Simulating resistive bridging and stuck-at faults. In *IEEE International Workshop On Current and Defect-Based Testing*, pages 49–56, 2003.
- [W2] P. Engelke, I. Polian, M. Renovell, and B. Becker. Simulating resistive bridging faults. In *GI/ITG Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen"*, pages 92–97, 2003.
- [W1] M. Keim, P. Engelke, and B. Becker. A parameterizable fault simulator for bridging faults. In *GI/ITG Workshop "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen"*, 2000. (*).

Bibliography

- [1] M. Abramovici, M.A. Breuer, and A.D. Friedman. *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.
- [2] M. Abramovici and P.R. Menon. A practical approach to fault simulation and test generation for bridging faults. *IEEE Trans. on Comp.*, C-34(7):658–663, July 1983.
- [3] J.M. Acken and S.D. Millman. Accurate modeling and simulation of bridging faults. In *Custom Integrated Circuits Conference*, pages 17.4.1–17.4.4, 1991.
- [4] J.M. Acken and S.D. Millman. Fault model evolution for diagnosis; accuracy vs precision. In *Custom Integrated Circuits Conference*, pages 13.4.1–13.4.4, 1992.
- [5] R.C. Aitken. Finding defects with fault models. In *Int'l Test Conf.*, pages 498–505, 1995.
- [6] R.C. Aitken. Defect or variation? Characterizing standard cell behavior at 90 nm and below. *IEEE Trans. on Semiconductor Manufacturing*, 21(1):46–54, Feb. 2008.
- [7] G.A. Allan and A.J. Walton. Hierarchical critical area extraction with the EYE tool. In *Int'l Workshop on Defect and Fault Tolerance in VLSI Systems*, pages 28–36, 1995.
- [8] M.E. Amyeen, S. Venkataraman, A. Ojha, and S. Lee. Evaluation of the quality of N-detect scan ATPG patterns on a processor. In *Int'l Test Conf.*, pages 669–678, 2004.
- [9] D. Arumí, R. Rodríguez-Montañés, J. Figueras, S. Eichenberger, C. Hora, B. Kruseman, M. Lousberg, and A.K. Majhi. Diagnosis of bridging defects based on current signatures at low power supply voltages. In *VLSI Test Symp.*, pages 145–150, 2007.
- [10] R.J. Baker. *CMOS – Circuit Design, Layout, and Simulation*. IEEE Press, 2nd edition, 1998.
- [11] P.H. Bardell, W.H. McAnney, and J. Savir. *Built In Test for VLSI: Pseudorandom Techniques*. John Wiley & Sons, New York, Dec. 1987.
- [12] B.Chess and C. Roth. On evaluating competing bridge fault models for CMOS ICs. In *VLSI Test Symp.*, pages 446–451, 1994.
- [13] B. Benware, C. Schuermyer, S. Ranganathan, R. Madge, P. Krishnamurty, N. Tamrapalli, K.H. Tsai, and J. Rajski. Impact of multiple-detect test patterns on product quality. In *Int'l Test Conf.*, pages 1031–1040, 2003.
- [14] B.R. Benware, R. Madge, C. Lu, and R. Daasch. Effectiveness comparisons of outlier screening methods for frequency dependent defects on complex ASICs. In *VLSI Test Symp.*, pages 39–46, 2003.

- [15] K.A. Bowman, X. Tang, J.C. Eble, and J.D Meindl. Impact of extrinsic and intrinsic parameter fluctuations on CMOS circuit performance. *IEEE Jour. of Solid-State Circ.*, 35(8):1186–1193, Aug. 2000.
- [16] D. Brand. Verification of large synthesized designs. In *Int'l Conf. on CAD*, pages 534–537, 1993.
- [17] R.K. Brayton, R. Rudell, A.L. Sangiovanni-Vincentelli, and A.R. Wang. MIS: A multiple - level logic optimization system. *IEEE Trans. on Comp.*, 6(6):1062–1081, 1987.
- [18] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Int'l Symp. Circ. and Systems*, pages 1929–1934, 1989.
- [19] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational circuits and a target translator in fortran. In *Int'l Symp. Circ. and Systems, Special Sess. on ATPG and Fault Simulation*, pages 663–698, 1985.
- [20] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.
- [21] M.L. Bushnell and V.D. Agrawal. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Kluwer Academic Publishers, 2001.
- [22] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu. New paradigm of predictive MOSFET and interconnect modeling for early circuit design. In *IEEE CICC*, pages 201–204, 2000.
- [23] K. Chakrabarty, B.T. Murray, and V. Iyengar. Built-in test pattern generation for high performance circuits using twisted-ring counters. In *VLSI Test Symp.*, pages 22–27, 1999.
- [24] S.T. Chakradhar, V.D. Agrawal, and S.G. Rothweiler. A Transitive Closure Algorithm for Test Generation. *IEEE Transactions on CAD*, 12:1015–1028, 1993.
- [25] S. Chakravarty, A. Jain, N. Radhakrishnan, E.W. Savage, and S.T. Zachariah. Experimental evaluation of scan tests for bridges. In *Int'l Test Conf.*, pages 509–518, 2002.
- [26] J.T.Y. Chang and E.J. McCluskey. Quantitative analysis of Very-Low-Voltage testing. In *VLSI Test Symp.*, pages 332–337, 1996.
- [27] T. Chen and I.N. Hajj. GOLDENGATE: a fast and accurate bridging fault simulator under a hybrid logic/IDDQ testing environment. In *Int'l Conf. on CAD*, pages 555–561, 1997.
- [28] B. Chess and T. Larrabee. Bridge fault simulation strategies for CMOS integrated circuits. In *Design Automation Conf.*, pages 458–462, 1993.
- [29] B. Chess and T. Larrabee. Logic testing of bridging faults in CMOS integrated circuits. *IEEE Trans. on Comp.*, 47(3):338–345, March 1998.
- [30] H. Cheung and S.K. Gupta. Accurate modeling and fault simulation of byzantine resistive bridges. In *Int'l Conf. on Comp. Design*, pages 347–353, 2007.

-
- [31] T. Clouqueur, K. Zarrineh, K.K. Saluja, and H. Fujiwara. Design and analysis of multiple weight linear compactors of responses containing unknown values. In *Int'l Test Conf.*, 2005.
- [32] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.
- [33] F. Corno, M.S. Reorda, and G. Squillero. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design & Test of Comp.*, 17:44–53, July/Sept. 2000.
- [34] J. P. Cusey and J. H. Patel. BART: A bridging fault test generator for sequential circuits. In *Int'l Test Conf.*, pages 838–847, 1997.
- [35] P. Dahlgren. Switch-level bridging fault simulation in the presence of feedbacks. In *Int'l Test Conf.*, pages 363–371, 1998.
- [36] M. Dalpasso, M. Favalli, P. Olivo, and B. Riccò. Parametric bridging fault characterization for the fault simulation of library-based ICs. In *Int'l Test Conf.*, pages 486–495, 1992.
- [37] C. Di and J.A.G. Jess. On CMOS bridge fault modeling and test pattern evaluation. In *VLSI Test Symp.*, pages 116–119, 1993.
- [38] C. Di and J.A.G. Jess. An efficient CMOS bridging fault simulator: With SPICE accuracy. *IEEE Trans. on CAD*, 15(9):1071–1080, Sept. 1996.
- [39] R. Drechsler and B. Becker. *Binary Decision Diagrams – Theory and Implementation*. Kluwer Academic Publishers, 1998.
- [40] E.B. Eichelberger and E. Lindbloom. Random-pattern coverage enhancement and diagnosis for LSSD logic self-test. *IBM J. Res. and Develop.*, 27(3):265–272, May 1983.
- [41] M. Favalli and M. Dalpasso. Symbolic handling of bridging fault effects. *Jour. of Electronic Testing: Theory and Applications*, 10:271–276, June 1997.
- [42] M. Favalli, P. Olivo, and B. Riccò. A probabilistic fault model for "analog" faults in digital CMOS circuits. *IEEE Trans. on CAD*, 11(11):1459–1462, Nov. 1992.
- [43] F.J. Ferguson and T. Larrabee. Test pattern generation for realistic bridge fault in CMOS ICs. In *Int'l Test Conf.*, pages 492–499, 1991.
- [44] F.J. Ferguson and J.P. Shen. A CMOS fault extractor for inductive fault analysis. *IEEE Trans. on CAD*, 7(11):1181–1194, Nov. 1988.
- [45] F.J. Ferguson and J.P. Shen. Extraction and simulation of realistic CMOS faults using inductive fault analysis. In *Int'l Test Conf.*, pages 475–484, 1988.
- [46] A.V. Ferris-Prabhu. Modeling the critical area in yield forecasts. *IEEE Jour. of Solid-State Circ.*, SC-20(4):874–878, Aug. 1985.
- [47] A.D. Friedman. Diagnosis of short-circuit faults in combinational circuits. *IEEE Trans. on Comp.*, C-23(7):746–752, July 1974.
- [48] A. Fudoli, A. Ascagni, D. Appello, and H. Manhaeve. A practical evaluation of IDDQ test strategies for deep submicron production test application. experiences and targets from the field. In *European Test Workshop*, pages 65–70, 2003.

- [49] H. Fujiwara and T. Shimono. On the acceleration of test generation algorithms. *IEEE Trans. on Comp.*, 32:1137–1144, 1983.
- [50] D.D. Gajski and R.H. Kuhn. Guest editors' introduction: New VLSI tools. *IEEE Computer*, 16(12):11–14, Dec. 1983.
- [51] J. Galiay, Y. Crouzet, and M. Vergniault. Physical versus logical fault models MOS LSI circuits: Impact on their testability. *IEEE Trans. on Comp.*, C-29(6):527–531, June 1980.
- [52] A.E. Gattiker and W. Maly. Current signatures. In *VLSI Test Symp.*, pages 112–117, 1996.
- [53] V. Gherman, H.-J. Wunderlich, H. Vranken, F. Hapke, M. Wittke, and M. Garbers. Efficient pattern mapping for deterministic logic BIST. In *Int'l Test Conf.*, pages 48–56, 2004.
- [54] E. Gizdarski and H. Fujiwara. SPIRIT: A highly robust combinational test generation algorithm. *IEEE Trans. on CAD*, 21(12):1446–1458, 12 2002.
- [55] M. Gkatziani, R. Kapur, Q. Su, B. Mathew, R. Mattiuzzo, L. Tarantini, C. Hay, S. Talluto, and T.W. Williams. Accurately determining bridging defects from layout. In *IEEE Int'l Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pages 87–90, 2007.
- [56] P. Goel. An implicit enumeration algorithm to generate test for combinational logic. *IEEE Trans. on Comp.*, 30:215–222, 1981.
- [57] F.M. Gonçalves, I.C. Teixeira, and J.P. Teixeira. Realistic fault extraction for high-quality design and test of VLSI systems. In *Int'l Symp. on Defect and Fault Tolerance in VLSI Systems*, pages 29–37, 1997.
- [58] M. Gössel. University of Potsdam. personal communication, 2008.
- [59] G.S. Greenstein. CMOS bridging fault simulation. Master's thesis, University of Illinois, Urbana-Champaign, 1992.
- [60] G.S. Greenstein and J.H. Patel. E-PROOFS: a CMOS bridging fault simulator. In *Int'l Conf. on CAD*, pages 268–271, 1992.
- [61] J.P. de Gyvez and C. Di. IC defect sensitivity for footprint-type spot defects. *IEEE Trans. on CAD*, 11(5):638–658, May 1992.
- [62] I. Hamzaoglu and J.H. Patel. New techniques for deterministic test pattern generation. *Jour. of Electronic Testing: Theory and Applications*, 15:63–73, 1999.
- [63] H. Hao and E.J. McCluskey. Very-Low-Voltage testing for weak CMOS logic ICs. In *Int'l Test Conf.*, pages 275–284, 1993.
- [64] M. Hashizume, N. Inou, H. Yotsuyanagi, and T. Tamesada. Oscillation frequency estimation for detecting feedback bridging faults. In *Int'l Tech. Conf. on Circuits/Systems, Comp. and Comm.*, pages 1980–1983, 2002.
- [65] M. Hashizume, H. Yotsuyanagi, and T. Tamesada. Identification of feedback bridging faults with oscillation. In *Asian Test Symp.*, pages 25–30, 1999.

-
- [66] C. Hawkins, J. Soden, A. Righter, and F. Joel Ferguson. Defect classes - an overdue paradigm for CMOS IC testing. In *Int'l Test Conf.*, pages 413–425, 1994.
- [67] J.P. Hayes. *Computer Architecture and Organization*. McGraw-Hill, 2nd edition, 1989.
- [68] J.P. Hayes and A.D. Friedman. Test point placement to simplify fault detection. *IEEE Trans. on Comp.*, C-33(7):727–735, 7 1974.
- [69] S. Hellebrand, H.G. Liang, and H.J. Wunderlich. A mixed-mode BIST scheme based on reseeding of folding counters. *Jour. of Electronic Testing: Theory and Applications*, 17(3-4):159–170, February 2001.
- [70] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois. Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift register. *IEEE Trans. on Comp.*, 44(2):223–233, February 1995.
- [71] S. Hellebrand, S. Tarnick, B. Courtois, and J. Rajski. Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift registers. In *Int'l Test Conf.*, pages 120–129, 1992.
- [72] P. Huc. *Test en tension des courts-circuits en technologie CMOS*. PhD thesis, Université de Montpellier II Sciences et Techniques du Languedoc, Montpellier, France, March 1995.
- [73] H. Iwai. CMOS technology – year 2010 and beyond. *IEEE Jour. of Solid-State Circ.*, 34(3):357–366, March 1999.
- [74] V.S. Iyengar, B.K. Rosen, and I. Spillinger. Delay test generation 1 – concepts and coverage metrics. In *Int'l Test Conf.*, pages 857–866, 1988.
- [75] A. Jee and F.J. Ferguson. Carafe: an inductive fault analysis tool for CMOS VLSI circuits. In *VLSI Test Symp.*, pages 92–98, 1993.
- [76] F. Jensen and N.E. Petersen. *Burn-in: an engineering approach to the design and analysis of burn-in procedures*. John Wiley & Sons, 1983.
- [77] N. Jha and S. Gupta. *Testing of Digital Systems*. Cambridge University Press, 2003.
- [78] B.W. Johnson. *The Design and Analysis of Fault Tolerant Digital Systems*. Addison Wesley, 1989.
- [79] R. Kapur, J. Park, and M.R. Mercer. All tests for a fault are not equally valuable for defect detection. In *Int'l Test Conf.*, pages 762–769, 1992.
- [80] M. Karpovsky and S.Y.H. Su. Detecting bridging and stuck-at faults at input and output pins of standard digital components. In *Design Automation Conf.*, pages 494–505, 1980.
- [81] M. Karpovsky and S.Y.H. Su. Detection and location of input and feedback bridging faults among input and output lines. *IEEE Trans. on Comp.*, C-29(6):523–527, June 1980.
- [82] A. Keshavarzi, K. Roy, C.F.Hawkins, and V. De. Multiple-parameter CMOS IC testing with increased sensitivity for IDDQ. *IEEE Trans. on VLSI Systems*, 11(5):863–870, Oct. 2003.

- [83] A. Keshavarzi, K. Roy, and C.F. Hawkins. Intrinsic leakage in low power deep submicron CMOS ICs. In *Int'l Test Conf.*, pages 146–155, 1997.
- [84] J. Khare and W. Maly. *From contamination to defects, faults and yield loss*. Kluwer Academic Publisher, 1996.
- [85] S. Khursheed, U. Ingelsson, P. Rosinger, B.M. Al-Hashimi, and P. Harrod. Bridging fault test method with adaptive power management awareness. *IEEE Trans. on CAD*, 27(6):1117 – 1127, June 2008.
- [86] G. Kiefer and H.-J. Wunderlich. Deterministic BIST with multiple scan chains. In *Int'l Test Conf.*, pages 1057–1064, 1998.
- [87] K.L. Kodandapani and D.K. Pradhan. Undetectability of bridging faults and validity of stuck-at fault test sets. *IEEE Trans. on Comp.*, C-29(1):55–59, Jan. 1980.
- [88] B. Könemann. LFSR-coded test patterns for scan designs. In *European Test Conf.*, pages 237–242, 1991.
- [89] H. Konuk and F. Joel Ferguson. Oscillation and sequential behavior caused by interconnect opens in digital CMOS circuits. In *Int'l Test Conf.*, pages 597–606, 1997.
- [90] A. Krasniewski and S. Pilarski. Circular self-test path: A low-cost BIST technique for VLSI circuits. *IEEE Trans. on CAD*, 8(1):46–55, 1989.
- [91] C.V. Krishna, A. Jas, and N.A. Touba. Test vector encoding using partial LFSR reseeding. In *Int'l Test Conf.*, pages 885–893, 2001.
- [92] V. Krishnaswamy, A.B. Ma, and P. Vishakantiah. A study of bridging defect probabilities on a Pentium(tm)4 CPU. In *Int'l Test Conf.*, pages 688–695, 2001.
- [93] B. Kruseman, S. van den Oetelaar, and J. Rius. Comparison of IDDQ testing and Very-Low Voltage testing. In *Int'l Test Conf.*, pages 964–973, 2002.
- [94] B. Kruseman, R. van Veen, and K. van Kaam. The future of delta-IDDQ testing. In *Int'l Test Conf.*, pages 101–110, 2001.
- [95] B. Krusemann and S. van den Oetelaar. Detection of resistive shorts in deep submicron technologies. In *Int'l Test Conf.*, pages 866–875, 2003.
- [96] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. on Programming Languages and Systems*, 4(3):382–401, July 1982.
- [97] T. Larrabee. Test pattern generation using boolean satisfiability. *IEEE Trans. on CAD*, 11:4–15, 1992.
- [98] C. Lee and D. M. H. Walker. PROBE: A PPSFP simulator for resistive bridging faults. In *VLSI Test Symp.*, pages 105–110, 2000.
- [99] K.-J. Lee and M.A. Breuer. Design and test rules for CMOS circuits to facilitate IDDQ testing of bridging faults. *IEEE Trans. on CAD*, 11(5):659–670, May 1992.
- [100] K.-J. Lee and J.-J. Tang. Two modeling techniques for CMOS circuits to enhance test generation and fault simulation for bridging faults. In *Asian Test Symp.*, pages 165–170, 1996.

-
- [101] M. Lewis, T. Schubert, and B. Becker. Multithreaded SAT solving. In *ASP Design Automation Conf.*, pages 926–931, 2007.
- [102] Z. Li, X. Lu, W. Qiu, W. Shi, and D.M.H. Walker. A circuit level fault model for resistive bridges. *ACM Trans. on Design Automation of Electronic Systems*, 8(4):546–559, 10 2003.
- [103] Y. Liao and D.M.H. Walker. Fault coverage analysis for physically-based CMOS bridging faults at different power supply voltages. In *Int'l Test Conf.*, pages 767–775, 1996.
- [104] Y. Liao and D.M.H. Walker. Optimal voltage testing for physically-based faults. In *Asian Test Symp.*, pages 344–353, 1996.
- [105] S. Ma, I. Shaik, and R.S. Fetherston. A comparison of bridging fault simulation methods. In *Int'l Test Conf.*, pages 587–595, 1999.
- [106] S.C. Ma, P. Franco, and E.J. McCluskey. An experimental chip to evaluate test techniques experimental results. In *Int'l Test Conf.*, pages 663–672, 1995.
- [107] T. Maeda and K. Kinoshita. Precise test generation for resistive bridging faults of CMOS combinational circuits. In *Int'l Test Conf.*, pages 510–519, 2000.
- [108] Y.K. Malaiya, A.P. Jayasumana, and R. Rajsuman. A detailed examination of bridging faults. In *Int'l Conf. on Comp. Design*, pages 78–81, 1986.
- [109] W. Maly. Realistic fault modeling for VLSI testing. In *Design Automation Conf.*, pages 173–180, 1987.
- [110] P. Maxwell, R. Aitken, and L. Huisman. The effect on quality of non-uniform fault coverage and fault probability. In *Int'l Test Conf.*, pages 739–746, 1994.
- [111] P. Maxwell, P. O'Neill, R. Aitken, R. Dudley, N. Jaarsma, M. Quach, and D. Wiseman. Current ratios: A self-scaling technique for production IDDQ testing. In *Int'l Test Conf.*, pages 738–746, 1999.
- [112] P.C. Maxwell and R.C. Aitken. Biased voting: A method for simulating CMOS bridging faults in the presence of variable gate logic thresholds. In *Int'l Test Conf.*, pages 63–72, 1993.
- [113] P.C. Maxwell and J.R. Rearick. Estimation of defect-free IDDQ in submicron circuits using switch level simulation. In *Int'l Test Conf.*, pages 882–889, 1998.
- [114] E.J. McCluskey. Built-in self-test techniques. *IEEE Design & Test of Comp.*, 2(2):21–28, April 1985.
- [115] K.C.Y. Mei. Bridging and stuck-at faults. *IEEE Trans. on Comp.*, C-23(7):720–727, July 1974.
- [116] S.M. Menon, A.P. Jayasumana, Y.K. Malaiya, and D.R. Clinkinbeard. Modelling and analysis of bridging faults in emitter-coupled logic (ECL) circuits. *IEE Proc. Computers and Digital Techniques*, 140(4):220–226, 1993.
- [117] N. Metropolis and S. Ulam. The Monte Carlo method. *Jour. of the Amer. Stat. Ass.*, 44(247):335–341, Sept. 1949.

- [118] S.F. Midkiff and S.W. Bollinger. Classification of bridging faults in CMOS circuits: experimental results and implications for test. In *VLSI Test Symp.*, pages 112–115, 1993.
- [119] A.C. Miller. IDDQ testing in deep submicron integrated circuits. In *Int'l Test Conf.*, pages 724–729, 1999.
- [120] S.D. Millman and J.M. Acken. Special applications of the voting model for bridging faults. *IEEE Jour. of Solid-State Circ.*, 29(3):263–270, March 1994.
- [121] S.D. Millman and Sir J.P. Garvey. An accurate bridging fault test pattern generator. In *Int'l Test Conf.*, pages 411–418, 1991.
- [122] S. Mitra and K.S. Kim. X-Compact: An efficient response compaction technique for test cost reduction. In *Int'l Test Conf.*, pages 311–320, 2002.
- [123] Y. Miura and S. Seno. Internal feedback bridging faults in combinational CMOS circuits: Analysis and testing. In *European Test Workshop*, pages 9–16, 2001.
- [124] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Design Automation Conf.*, 2001.
- [125] P.K. Nag and W. Maly. Hierarchical extraction of critical area for shorts in very large ICs. In *IEEE Int'l Workshop on Current and Defect-Based Testing*, pages 19–27, 1995.
- [126] M. Naruse, I. Pomeranz, S.M. Reddy, and S. Kundu. On-chip compression of output responses with unknown values using LFSR reseeding. In *Int'l Test Conf.*, pages 1060–1068, 2003.
- [127] S.R. Nassif. Modeling and analysis of manufacturing variations. In *Custom Integrated Circuits Conference*, pages 11.1.1–11.1.6, 2001.
- [128] W. Needham, C. Prunty, and E.H. Yeoh. High volume microprocessor test escapes, an analysis of defects our tests are missing. In *Int'l Test Conf.*, pages 25–34, 1998.
- [129] P. Nigh and W. Maly. Layout-driven test generation. In *Int'l Conf. on CAD*, pages 154–157, 1989.
- [130] E. Papadopoulou and D.T. Lee. Critical area computation via Voronoi diagrams. *IEEE Trans. on CAD*, 18(4):463–474, April 1999.
- [131] J. Park, M. Naivar, R. Kapur, M.R. Mercer, and T.W. Williams. Limitations in predicting defect level based on stuck-at fault coverage. In *VLSI Test Symp.*, pages 186–191, 1994.
- [132] R. Paul. *Elektrotechnik: Grundlagenlehrbuch, Bd. 1. Felder und einfache Stromkreise*. Springer-Verlag, 3rd edition, 1993.
- [133] M.G. Pecht, R. Radojic, and G. Rao. *Managing Silicon Chip Reliability*. CRC Press, 1998.
- [134] F. Peters and S. Oostdijk. Realistic defect coverages of voltage and current tests. In *Int'l Workshop on IDDQ Testing*, pages 4–8, 1996.

-
- [135] S. Pilarski and A. Perzyńska. BIST and delay fault detection. In *Int'l Test Conf.*, pages 236–242, 1993.
- [136] I. Polian. *On Non-standard Fault Models for Logic Digital Circuits: Simulation, Design for Testability, Industrial Applications*. VDI-Verlag, Düsseldorf, vdi fortschritt-berichte edition, March 2004.
- [137] I. Polian and B. Becker. Scalable delay fault BIST for use with low-cost ATE. *Jour. of Electronic Testing: Theory and Applications*, 20(2):181–197, 4 2004.
- [138] I. Pomeranz, S. Kundu, and S.M. Reddy. On output response compression in the presence compression in the response of unknown output values. In *Design Automation Conf.*, pages 255–258, 2002.
- [139] T.J. Powell, J. Pair, M. St. John, and D. Counce. Delta IDDQ for testing reliability. In *VLSI Test Symp.*, pages 439–443, 2000.
- [140] J. Rajski and J. Tyszer. Synthesis of X-tolerant convolutional compactors. In *VLSI Test Symp.*, pages 114–119, 2005.
- [141] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee. Embedded deterministic test. *IEEE Trans. on CAD*, 23(5):776–792, 5 2004.
- [142] J. Rajski, C. Wang, J. Tyszer, and S.M. Reddy. Convolutional compaction of test responses. In *Int'l Test Conf.*, pages 745–754, 2003.
- [143] R. Rajsuman. An analysis of feedback bridging faults in MOS VLSI. In *VLSI Test Symp.*, pages 53–58, 1991.
- [144] R. Rajsuman. IDDQ testing for CMOS VLSI. *Proc. of the IEEE*, 88(4):544–568, April 2000.
- [145] R. Rajsuman, Y.K. Malaiya, and A.P. Jayasumana. On accuracy of switch-level modeling of bridging faults in complex gates. In *Design Automation Conf.*, pages 244–250, 1987.
- [146] J. Rearick and J.H. Patel. Fast and Accurate CMOS Bridging Fault Simulation. In *Int'l Test Conf.*, pages 54–62, 1993.
- [147] S.M. Reddy, I. Pomeranz, and S. Kajihara. Compact test sets for high defect coverage. *IEEE Trans. on CAD*, 16(8):923–930, Aug. 1997.
- [148] S.M. Reddy, I. Pomeranz, H. Tang, S. Kajihara, and K. Kinoshita. On testing of interconnect open defects in combinational logic circuits with stems of large fanout. In *Int'l Test Conf.*, pages 83–89, 2002.
- [149] M. Renovell, F. Azais, and Y. Bertrand. Detection of defects using fault model oriented test sequences. *Jour. of Electronic Testing: Theory and Applications*, 14:13–22, 1999.
- [150] M. Renovell, F. Azais, and Y. Bertrand. Improving defect detection in static-voltage testing. *IEEE Design & Test of Comp.*, 19(6):32–38, Nov./Dec. 2002.
- [151] M. Renovell and Y. Bertrand. Test strategy sensitivity to defect parameters. In *Int'l Test Conf.*, pages 607–616, 1997.

- [152] M. Renovell, P. Huc, and Y. Bertrand. CMOS bridge fault modeling. In *VLSI Test Symp.*, pages 392–397, 1994.
- [153] M. Renovell, P. Huc, and Y. Bertrand. A unified model for inter-gate and intra-gate CMOS bridging fault: the configuration ratio. In *Asian Test Symp.*, pages 170–175, 1994.
- [154] M. Renovell, P. Huc, and Y. Bertrand. The concept of resistance interval: A new parametric model for resistive bridging fault. In *VLSI Test Symp.*, pages 184–189, 1995.
- [155] M. Renovell, P. Huc, and Y. Bertrand. Serial transistor network modeling for bridging fault simulation. In *Asian Test Symp.*, pages 100–106, 1995.
- [156] M. Renovell, P. Huc, and Y. Bertrand. Bridging fault coverage improvement by power supply control. In *VLSI Test Symp.*, pages 338–343, 1996.
- [157] M.L. Rieger, J.P. Mayhew, and S. Panchapakesan. Layout design methodologies for sub-wavelength manufacturing. In *Design Automation Conf.*, pages 85–88, 2001.
- [158] R. Rodríguez-Montañés, E.M.J.G. Bruls, and J. Figueras. Bridging defects resistance measurements in a CMOS process. In *Int'l Test Conf.*, pages 892–899, 1992.
- [159] R. Rodríguez-Montañés and J. Figueras. Analysis of bridging defects in sequential CMOS circuits and their current testability. In *European Design & Test Conf.*, pages 356–360, 1994.
- [160] R. Rodríguez-Montañés, J. Figueras, and A. Rubio. Current vs. logic testability of bridges in scan chains. In *European Test Conference*, pages 392–396, 1993.
- [161] J.P. Roth. Diagnosis of automata failures: A calculus and a method. *IBM J. Res. Dev.*, 10:278–281, 1966.
- [162] A. Rubio, J. Figueras, V. Champac, R. Rodríguez, and J. Segura. IDDQ secondary components in CMOS logic circuits preceded by defective stages affected by analogue type faults. *IEE Electronics Letters*, 27(18):1656–1658, Aug. 1991.
- [163] M. Sachdev and J.P. de Gyvez. *Defect-Oriented Testing for Nano-Metric CMOS VLSI Circuits*. Springer, Dordrecht, 2nd edition, 2007.
- [164] T. Sakurai and A.R. Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *IEEE Jour. of Solid-State Circ.*, 25(2):584–594, 4 1990.
- [165] V. Sar-Dessai and D.M.H. Walker. Accurate fault modeling and fault simulation of resistive bridges. In *Int. Symp. Defect and Fault Tolerance in VLSI Systems*, pages 102–107, 1998.
- [166] V. Sar-Dessai and D.M.H. Walker. Resistive Bridge Fault Modeling, Simulation and Test Generation. In *Int'l Test Conf.*, pages 596–605, 1999.
- [167] M.H. Schulz and D. Pellkofer. A Three-Valued Fast Fault Simulator for Scan-Based VLSI-Logic. In *European Test Conference*, pages 41–48, Apr. 1989.
- [168] M.H. Schulz, E. Trischler, and T.M. Sarfert. SOCRATES: A highly efficient automatic test pattern generation system. *IEEE Trans. on CAD*, 7(1):126–137, Jan. 1988.

-
- [169] S. Sengupta, S. Kundu, S. Chakravarty, P. Paravathala, R. Galivanche, G. Kosonocky, M. Rodgers, and TM Mak. Defect-based test: A key enabler for successful migration to structural test. *Intel Technology Journal*, 1, 1999.
- [170] B. Seshadri, I. Pomeranz, S.M. Reddy, and S. Kundu. On path selection for delay fault testing considering operating conditions. In *European Test Workshop*, pages 141–146, 2003.
- [171] S. Seshu. On an improved diagnosis program. *IEEE Trans. on Electronic Comp.*, 12(2):76–79, 1965.
- [172] J.P. Shen, W. Maly, and F.J. Ferguson. Inductive Fault Analysis of MOS integrated circuits. *IEEE Design & Test of Comp.*, 2(6):13–26, Dec. 1985.
- [173] H.-C. Shih and J.A. Abraham. Transistor-level test generation for physical failures in CMOS circuits. In *Design Automation Conf.*, pages 243–249, 1986.
- [174] T. Shinogi, T. Kanbayashi, T. Yoshikawa, S. Tsuruoka, and T. Hayashi. Faulty resistance sectioning technique for resistive bridging fault ATPG systems. In *Asian Test Symp.*, pages 76–81, 2001.
- [175] W. Shockley and G.L. Pearson. Modulation of conductance of thin films of semiconductors by surface charges. *Physical Review*, 74:232, 1948.
- [176] J.P. Marques Silva and K.A. Sakallah. Robust search algorithms for test pattern generation. In *Int'l Symp. on Fault-Tolerant Comp.*, pages 152–161, 1997.
- [177] J.J.T. Sousa, F.M. Gonçalves, and J.P. Teixeira. IC defects-based testability analysis. In *Int'l Test Conf.*, pages 500–509, 1991.
- [178] M. Spica, M. Tripp, and R. Roeder. A new understanding of bridge defect resistances and process interactions from correlating inductive fault analysis predictions to empirical test results. In *IEEE Int'l Workshop on Current and Defect-Based Testing*, pages 11–16, 2001.
- [179] Z. Stanojevic and D.M.H. Walker. FedEx – a fast bridging fault extractor. In *Int'l Test Conf.*, pages 696–703, 2001.
- [180] C. Stapper. Modeling of integrated circuit defect sensitivities. *IBM J. Res. and Develop.*, 27(6):549–557, Nov. 1983.
- [181] P. Stephan, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. Combinational test generation using satisfiability. *IEEE Trans. on CAD*, 15(9):1167–1176, Sept. 1996.
- [182] C.E. Stroud, J.M. Emmert, J.R. Bailey, K.S. Chhor, and D. Nikolic. Bridging fault extraction from physical design data for manufacturing test development. In *Int'l Test Conf.*, pages 688–695, 2000.
- [183] P. Thadikaran, S. Chakravarty, and J. Patel. Fault simulation of I_{DDQ} tests for bridging faults in sequential circuits. In *Int'l Symp. on Fault-Tolerant Comp.*, pages 340–349, 1995.
- [184] C. Thibeault. A novel probabilistic approach for IC diagnosis based on differential quiescent current signatures. In *VLSI Test Symp.*, pages 80–85, 1997.

- [185] C. Thibeault. On the comparison of Delta Iddq and Iddq testing. In *VLSI Test Symp.*, pages 143–150, 1999.
- [186] N.A. Touba and E.J. McCluskey. Transformed pseudo-random patterns for BIST. In *VLSI Test Symp.*, pages 410–416, 1995.
- [187] N.A. Touba and E.J. McCluskey. Altering a pseudo-random bit sequence for scan based bist. In *Int'l Test Conf.*, pages 649–658, 1996.
- [188] C.-W. Tseng, R. Chen, P. Nigh, and E.J. McCluskey. MINVDD testing for weak CMOS ICs. In *Asian Test Symp.*, pages 339–344, 2001.
- [189] C.W. Tseng, E.J. McCluskey, X. Shao, J. Wu, and D.M. Wu. Cold delay defect screening. In *VLSI Test Symp.*, pages 183–188, 2000.
- [190] H. Vierhaus, W. Meyer, and U. Glaser. CMOS bridges and resistive transistor faults: IDDQ versus delay effects. In *Int'l Test Conf.*, pages 83–91, 1993.
- [191] E.H. Volkerink and S. Mitra. Response compaction with any number of unknowns using a new LFSR architecture. In *Design Automation Conf.*, pages 117–122, 2005.
- [192] K.D. Wagner and E.J. McCluskey. Effect of supply voltage on circuit propagation delay and test applications. In *Int'l Conf. on CAD*, pages 42–44, 1985.
- [193] J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and T. McCarthy. Fault simulation for structured VLSI. *VLSI Systems Design*, 6(12):20–32, Dec. 1985.
- [194] J.A. Waicukauski, P.A. Shupe, D.J. Giramma, and A. Matin. ATPG for Ultra-Large Structured Designs. In *Int'l. Test Conference*, pages 44–51, 1990.
- [195] D.M.H. Walker and S.W. Director. VLASIC: A catastrophic fault yield simulator for integrated circuits. *IEEE Trans. on CAD*, CAD-5(4):541–556, Oct. 1986.
- [196] R.A. Walker and D.E. Thomas. A model of design representation and synthesis. In *Design Automation Conf.*, pages 453–459, 1985.
- [197] L. Wang, S.K. Gupta, and M.A. Breuer. Modeling and simulation for crosstalk aggravated by weak-bridge defects between on-chip interconnects. In *Asian Test Symp.*, pages 440–447, 2004.
- [198] S. Wang, W. Wei, and S.T. Chakradhar. Unknown blocking scheme for low control data volume and high observability. In *Design, Automation and Test in Europe*, pages 33–38, 2007.
- [199] W. Wang and S.K. Gupta. Weighted random robust path delay testing of synthesized multilevel circuits. In *VLSI Test Symp.*, pages 291–297, 1994.
- [200] F. Wanlass and C. Sah. Nanowatt logic using field-effect metal-oxide semiconductor triodes. In *ISSCC Digest of Technical Papers*, pages 32–33, 1963.
- [201] N.H.E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design : a systems perspective*. Addison-Wesley, 2nd edition, 1994.
- [202] M.J.Y. Williams and J.B. Angell. Enhancing testability of large-scale integrated circuits via test points and additional logic. *IEEE Trans. on Comp.*, C-22(1):46–60, Jan. 1973.

-
- [203] T.W. Williams, R.H. Dennard, R. Kapur, M.R. Mercer, and M. Maly. IDDQ test: Sensitivity analysis of scaling. In *Int'l Test Conf.*, pages 786–792, 1996.
- [204] M.H. Woods. MOS VLSI reliability and yield trends. *Proc. of the IEEE*, 74(12):1715–1729, 1986.
- [205] H.-J. Wunderlich. On computing optimized input probabilities for random tests. In *Design Automation Conf.*, pages 392–398, Oct. 1987.
- [206] H.-J. Wunderlich and G. Kiefer. Bit-flipping BIST. In *Int'l Conf. on CAD*, pages 337–343, 1996.
- [207] B. Wurth and K. Fuchs. A BIST Approach to Delay Fault Testing with Reduced Test Length. In *European Design & Test Conf.*, pages 418–423, 1995.
- [208] X. Xi, M. Dunga, J. He, W. Liu, K.M. Cao, X. Jin, J.J. Ou, M. Chan, A.M. Niknejad, C. Hu, and Ali Niknejad. *BSIM4.4.0 MOSFET Model – User's Manual*. Dept. of Electrical Engineering and Computer Sciences UC Berkeley, 2004.
- [209] S. Xu and S.Y.H. Su. Testing feedback bridging faults among internal, input and output lines by two patterns. In *Int'l Conf. on Circuits and Computers*, pages 214–217, 1982.
- [210] S. Xu and S.Y.H. Su. Detecting I/O and internal feedback bridging faults. *IEEE Trans. on Comp.*, C-34(6):553–557, June 1985.
- [211] H. Xue, C. Di, and J.A.G. Jess. A net-oriented method for realistic fault analysis. In *Int'l Conf. on CAD*, pages 78–83, 1993.
- [212] N. Zacharia, J. Rajski, and J. Tyszer. Decompression of test data using using variable-length seed LFSRs. In *VLSI Test Symp.*, pages 426–433, 1995.
- [213] S.T. Zachariah and S. Chakravarty. Extraction of two-node bridges from large industrial circuits. *IEEE Trans. on CAD*, 23(3):433–439, March 2004.
- [214] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45nm early design exploration. *IEEE Trans. on Electron Devices*, 53(11):2816–2823, Nov. 2006.

List of Algorithms

5.1	Computation of $V_{n_0} = I_n^{-1}(I_0)$ for a single (or parallel) n -transistor(s).	57
5.2	Computation of $I_0 = I_n(V_{n_0})$ for two n -transistor in series.	58
5.3	Computation of $V_{n_0} = I_n^{-1}(I_0)$ for two n -transistors in series.	59
5.4	Generation of sorted list of critical resistances R_{crit} for bridging fault f	69
7.1	PPSFP simulation of sections from S for test pattern set P	90
8.1	ATPG procedure for resistive bridging faults.	110
10.1	Monte Carlo estimation of non-target defect coverage.	184

List of Figures

2.1	Example of (a) combinational and (b) sequential circuit.	7
2.2	Model of a synchronous sequential circuit S	8
2.3	Symbol of (a) p -channel and (b) n -channel MOS-FET.	9
2.4	Standard CMOS implementation of a NOT gate with input A and output Y	10
2.5	Pull-up and pull-down network of a logic gate.	11
2.6	Time-frame expansion of a synchronous sequential circuit.	15
2.7	Basic built-in self test architecture.	18
3.1	Example circuit with a bridging fault.	23
3.2	Portion of the circuit relevant for bridging fault models.	24
3.3	Influence of adjacency region and lateral distance on capacitance.	26
4.1	Transistor level view of driving gates in CMOS technology.	34
4.2	Interpretation of voltage V_{br} at the bridged nodes a and b by driven gates C and D	37
4.3	Example of circuit with a non-resistive bridging fault.	39
4.4	Primary outputs reachable from driven gates' inputs.	40
5.1	Example of circuit with a resistive short affecting nodes a and b	46
5.2	Characteristics of voltages at nodes a and b as a function of the short's resistance R_{sh} : (a) input assignment $(0, 0, 1, 1)$, and (b) input assignments $(0, 0, 1, 1)$ and $(0, 1, 1, 1)$	47
5.3	Transistor level view of a resistive bridging fault shorting two inverters.	50
5.4	Transistor level view of a series network with (a) two n -transistors, and (b) two p -transistors.	57
5.5	Probability density function ρ for short resistance r as proposed by Renovell et al. [154].	61
5.6	Example of interval-based simulation.	67
5.7	Prototype of a two successor bridge affecting nodes a and b	76
5.8	Possible voltage characteristics and critical resistances for activating assignments: (a) $a = 1, b = 0$, and (b) $a = 0, b = 1$	77
7.1	Resistive bridging fault simulation flow.	86
7.2	Data structure for a single bridging fault.	88
7.3	Simulation time per resistive bridging fault as function of circuit size.	94
7.4	Simulation time for resistive bridging faults vs. single-stuck-at faults.	94
7.5	Run-time of SUPERB vs. interval-based approach from [J3].	95

8.1	Example of circuit with a resistive short affecting nodes a and b	107
8.2	Characteristics of voltages at nodes a and b as a function of short resistance R_{sh} : (a) input assignment (1, 0) and (b) input assignment (0, 1).	108
8.3	Performance of resistive bridging fault test vectors in comparison with combination of stuck-at and top-up vectors.	114
8.4	Number of test vectors (bar graph) and share of test vectors effective in detection of resistive bridging faults (line graph) for n -detection and 4-way test sets.	118
8.5	Resistive bridging fault coverage of n -detection and 4-way test sets.	119
8.6	Number of test vectors for different technology models. Horizontal line indicates average over all circuits/technology models considered in Table 8.8.	120
9.1	Example of circuit with a resistive short affecting nodes a and b	130
9.2	Voltage characteristic at node a as a function of R_{sh} for nominal (“nom”) and non-nominal (“nn”) operating conditions and two test vectors.	131
9.3	Venn diagram for non-nominal fault coverage FC^{nn} (Equation (9.1.1)). Diagonal lines indicate the numerator, vertical lines show the denominator.	133
9.4	Venn diagram for combined fault coverage FC_{comb}^{nn} (Equation (9.1.2)). Diagonal lines indicate the numerator, vertical lines show the denominator.	133
9.5	Venn diagram for flaw coverage FC_{flaw}^{nn} (Equation (9.1.3)). Diagonal lines indicate the numerator, vertical lines show the denominator.	134
9.6	Performance degradation due to low-voltage testing.	135
9.7	FC^{nn} for c2670 and different values of V_{dd} as function of test time for time units 0 through 100.	139
9.8	FC^{nn} for c2670 and different values of V_{dd} as function of test time for time units 100 through 1,000.	139
9.9	Average flaw coverage FC_{flaw}^{nn} from Tables 9.8 – 9.11.	142
9.10	Example circuit with a resistive short affecting nodes a and b	143
9.11	Voltage characteristics at nodes a and b as a function of R_{sh} for nominal (“nom”) and non-nominal (“nn”) operating conditions.	144
9.12	Venn diagram for coverage loss FC_{loss}^{nn} (Equation (9.1.5)). Diagonal lines indicate the numerator, vertical lines show the denominator.	145
9.13	Example of a sequential circuit with a resistive short affecting nodes a and b .	148
9.14	Current flowing through the defect (upper diagram) and voltage characteristic at nodes a and b (lower diagram) as a function of R_{sh} for two test vectors.	149
9.15	Venn diagram for (a) combined fault coverage FC_{comb}^{IDDQ} (Equation (9.2.4)) and (b) flaw coverage FC_{flaw}^{IDDQ} (Equation (9.2.5)). Diagonal lines indicate the numerator, vertical lines show the denominator.	152
9.16	Two-frame time-frame expansion of the circuit from Figure 9.13 (I -ADIs are marked by “ $IDDQ$ ”).	153
9.17	Detection conditions for the second time-frame. Resistance ranges highlighted in (light) gray are detected by Delta-IDDQ testing.	154
9.18	IDDQ fault coverage FC^{IDDQ} (bar graph) and number of AA faults (line graph) for 10, 100, and 1,000 Delta-IDDQ measurements.	156

9.19	IDDQ fault coverage FC^{IDDQ} for Delta-IDDQ and IDDQ testing (10 IDDQ measurements).	157
10.1	Schematic of circuit under test with bit-flipping logic.	174
10.2	Average fault coverage G -FE and average logic size from Table 10.2.	178
10.3	Deterministic logic BIST with X-masking logic.	181
10.4	Results for c1355, pattern-based relevant bit selection and 1% Xs: Number of masked bits and logic size as function of n (a); RBF coverage and logic size as function of n (b).	186
10.5	Results for c1355, bit-based relevant bit selection and 1% Xs: Number of masked bits and logic size as function of n (a); RBF coverage and logic size as function of n (b).	186
10.6	Results for c1355, bit-based relevant bit selection and 3% Xs: Number of masked bits and logic size as function of n (a); RBF coverage and logic size as function of n (b).	187

List of Tables

2.1	Symbols of different types of logic gates.	6
4.1	Aggressor and victim nodes for wired-logic and dominance-behavior models, “X” marks combination of aggressor, victim, and stuck-at fault used by the model.	31
4.2	Aggressor and victim nodes for 4-way model.	33
4.3	Instances of unified fault model.	41
5.1	Naming conventions for the difference in the voltage potentials between the transistor terminals.	51
5.2	Technological parameters for Shockley and Fitted model (n - and p -channel MOS-FET).	52
5.3	Results of Shockley and Fitted models compared to HSPICE; $L_p = L_n = 0.35 \mu\text{m}$, $V_{\text{dd}} = 3.3 \text{V}$, logic threshold $V_{\text{it}} = V_{\text{dd}}/2$	56
5.4	Resistance intervals reconverging at XOR gate.	65
5.5	Section-based propagation in circuit from Figure 5.6.	71
5.6	Conditional multiple-stuck-at faults corresponding to resistive bridging faults.	72
7.1	Comparison with the parallel-pattern resistive bridging fault simulator from Cheung et al. [30].	96
7.2	Experimental results for SUPERB, ISCAS 85 circuits, and 10,000 test vectors.	98
7.3	Experimental results for SUPERB, combinational cores of ISCAS 89 circuits, and 10,000 test vectors.	99
7.4	Experimental results for SUPERB, combinational cores of ITC 99 circuits, and 10,000 test vectors.	100
7.5	Experimental results for SUPERB, combinational cores of NXP circuits, and 10,000 test vectors.	101
7.6	Comparison of SUPERB with other simulation tools for resistive bridging faults.	102
8.1	Section data for resistive bridging fault shorting a two-input NAND and a two-input NOR gate. Symbol “X” indicates that any driven gate interprets a faulty logical value.	106
8.2	Efficiency of RBF-ATPG compared with BART and ATPG by Sar-Dessai et al.	115
8.3	Efficiency of RBF-ATPG compared with unified model (instance ISFN_TH).	116
8.4	Experimental results for RBF-ATPG.	122
8.5	Performance of single-stuck-at test sets.	123

8.6	Efficiency comparison with Resistive Bridging Fault Test Generator RBFTG [107] and single-stuck-at ATPGs ATOM [62] and SPIRIT [54].	124
8.7	Test vector sets for n -detection and 4-way model and their resistive bridging fault coverage.	125
8.8	Comparison of ATPG results for different technology models.	126
9.1	Average FC_{comb}^{nn} for different values of V_{dd} , T^{nom} , and T	141
9.2	Coverage loss FC_{loss}^{nn} (circuits with $FC_{loss}^{nn} = 0\%$ for all V_{dd} settings excluded).	145
9.3	Non-nominal fault coverage FC^{nn} for 10 time units and $V_{dd}^{nom} = 3.3$ V.	159
9.4	Non-nominal fault coverage FC^{nn} for 100 time units and $V_{dd}^{nom} = 3.3$ V.	160
9.5	Non-nominal fault coverage FC^{nn} for 1,000 time units and $V_{dd}^{nom} = 3.3$ V.	161
9.6	Flaw coverage FC_{flaw}^{nn} for 1,000 time units.	162
9.7	Combined fault coverage FC_{comb}^{nn} for $T^{nom} = 370$ K, $T = 300$ K, $V_{dd}^{nom} = 3.3$ V and different values of V_{dd}	163
9.8	Flaw coverage FC_{flaw}^{nn} for $V_{dd}^{nom} = 3.3$ V, $T^{nom} = T = 370$ K and different values of V_{dd}	164
9.9	Flaw coverage FC_{flaw}^{nn} for $V_{dd}^{nom} = 3.3$ V, $T^{nom} = 370$ K, $T = 300$ K and different values of V_{dd}	165
9.10	Flaw coverage FC_{flaw}^{nn} for $V_{dd}^{nom} = 3.3$ V, $T^{nom} = 370$ K, $T = 196$ K and different values of V_{dd}	166
9.11	Flaw coverage FC_{flaw}^{nn} for $V_{dd}^{nom} = 3.3$ V, $T^{nom} = 300$ K, $T = 196$ K and different values of V_{dd}	167
9.12	Fault coverages for Delta-IDDQ testing with 1,000 test vectors for voltage testing, current threshold $\Delta I_{limit} = 100 \mu A$ and 10 IDDQ measurements.	168
9.13	Fault coverages for Delta-IDDQ testing with 1,000 test vectors for voltage testing, current threshold $\Delta I_{limit} = 100 \mu A$ and 100 IDDQ measurements.	169
9.14	Fault coverages for Delta-IDDQ testing with 1,000 test vectors for voltage testing, current threshold $\Delta I_{limit} = 100 \mu A$ and 1,000 IDDQ measurements.	170
10.1	Stuck-at coverage of pseudo-random sequences before deterministic pattern embedding.	176
10.2	Fault coverage G -FE and logic size in GE for sequences of 1,000, 5,000, and 10,000 patterns.	177
10.3	Experimental results, pattern-based relevant bit selection (1% X values at the inputs).	189
10.4	Experimental results, bit-based relevant bit selection (1% X values at the inputs).	190
10.5	Experimental results, bit-based relevant bit selection (3% X values at the inputs).	191