Fault Tolerance Infrastructure and its Reuse for Offline Testing

Synergies of a Unified Architecture to Cope with Soft Errors and Hard Faults

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Michael E. Imhof

aus Kirchheim unter Teck

Hauptberichter: Prof. Dr. rer. nat. habil. Hans-Joachim Wunderlich
Mitberichter: Prof. Dr. rer. nat. habil. Sybille Hellebrand

Tag der mündlichen Prüfung: 15. September 2015

Institut für Technische Informatik der Universität Stuttgart
2015

To my family.

Acknowledgments

It is my pleasure to thank those who accompanied me on my doctoral journey.

I am very grateful to my parents, Karola and Otwin Imhof, and my sister Birgit, for their enduring support and encouragement throughout my education and studies. Without them, I would not have been able to even start with a work like this.

I would like to thank Prof. Hans-Joachim Wunderlich for his professional supervision and for the constructive feedback he provided during this dissertation. He enriched my time at the institute towards becoming an independent researcher with opportunities and challenges for professional and personal development that made the past years so much more than just writing a thesis. I would also like to thank Prof. Sybille Hellebrand for her support and for accepting to be the second adviser of my thesis.

I very much enjoyed the work with colleagues and students who were at some time involved in my activities in Stuttgart. Sorry, I cannot list everyone here, you are not forgotten. I would like to acknowledge those with whom I spend a lot of time in collaborations or personally: Rafał Baranowski, Günter Bartsch, Lars Bauer, Claus Braun, Alejandro Cook, Alexander Czutro, Atefe Dalirsani, Rainer Dorsch, Marcus Eggenberger, Melanie Elm, Laura Rodríguez Gómez, Nadereh Hatami, Stefan Holst, Manuel Jerger, Rauf Salimi Khaligh, Michael Kochte, Chang Liu, Abdullah Mumtaz, Matthias Sauer, Gert Schley, Eric Schneider, Hongyan Zhang, and Christian Zöllin.

Such a work is not possible without administrative and technical assistance: Thank you Mirjam Breitling, Helmut Häfner, Lothar Hellmeier, and Wolfgang Moser.

Stuttgart, September 2015

Michael E. Imhof

Contents

Acknowledgments iii

Contents v

List of Figures xi

List of Tables xv

Acronyms xvii

Notation xix

Summary xxi

Zusammenfassung xxiii

1. Introduction 1
- 1.1. Failure Mechanisms in Integrated Circuits 3
  - 1.1.1. Permanent Faults in CMOS Materials 4
  - 1.1.2. Radiation-induced Soft Errors in CMOS Transistors 6
- 1.2. Test and Design for Test 12
  - 1.2.1. Testability and Test Infrastructure 13
  - 1.2.2. Test Economics 14
- 1.3. Soft Error Mitigation 15
- 1.4. Overview and Contributions 17

Part I – Formal Foundation and Related Work 21

2. Formal Foundation 23

2.1. Digital Circuits 23
2.1.1. Modeling Levels 23
2.1.2. Combinational Circuit 24
2.1.3. Sequential Circuit 26
2.1.4. Defect, Fault, Error, Failure 26
2.2. Soft Errors 28
2.2.1. Used Soft Error Nomenclature 28
2.2.2. Soft Error Quantification 29
2.2.3. Fault Tolerance 30
2.3. Test of Digital Circuits 33
2.3.1. Fault Models 33
2.3.2. Test Access through Scan Design 35
2.3.3. Test Algorithms 37
2.4. Boolean Satisfiability 39

3. Related Work in Soft Error Mitigation and Test Access 41

3.1. Soft Error Mitigation 41
3.1.1. Dedicated Memories 41
3.1.2. Sequential Elements in Random Logic 45
3.2. Test Access 51
3.2.1. Test Data Compression and Compaction 51
3.2.2. Random Access Scan 53
3.3. Combined Solutions 56

Part II – Fault Tolerance Infrastructure 59

4. Non-Concurrent Detection and Localization of Single Event Upsets 61

4.1. Non-Concurrent Architecture 62
4.2. Single Event Upset Detection at Gate Level 63
4.2.1. Register Parity Protection 64
4.2.2. Area Efficient Register Parity Computation - Parity-Pair Latch 66
4.3. Single Event Upset Localization at Module Level 68
4.3.1. Modulo-2 Address Characteristic 68
4.3.2. Optimal Combinational Characteristic Computation 71

4.4. Experimental Evaluation 74
4.4.1. Experimental Setup 74
4.4.2. Single Event Upset Detection at Gate Level 75
4.4.3. Single Event Upset Localization at Module Level 81
4.5. Summary 83

5. Concurrent Online Correction of Single Event Upsets 85
5.1. Concurrent Online Architecture 86
5.2. Single Error Detection (SED) 87
5.2.1. Derivation of a Register Specific Error Condition 88
5.2.2. Protected Storage of the Error Condition 89
5.3. Single Error Correction (SEC) 90
5.3.1. Rapid Correction by Bit-Flipping 90
5.3.2. Timing Behavior of the Online Correction 92
5.4. Experimental Evaluation 92
5.4.1. Experimental Setup 93
5.4.2. Single Error Detection (SED) 94
5.4.3. Single Error Correction (SEC) 95
5.5. Summary 101

6. Fault Tolerance in Presence of Multiple Bit Upsets 103
6.1. Preliminary Error Multiplicity Considerations 104
6.2. Online Architecture for Double Errors 105
6.3. Optimal Extended Characteristic Computation 106
6.4. Experimental Evaluation 108
6.4.1. Experimental Setup 108
6.4.2. Area Overhead 108
6.5. Summary 110

7. Area Efficient Characteristic Computation 111
7.1. Detailed Analysis of the Correction Area Overhead 112
7.2. Area Efficient Exclusive OR Trees 112
7.3. Experimental Evaluation 115
7.3.1. Experimental Setup 115
7.3.2. Transmission-Gate Exclusive OR Standard Cell 115
7.3.3. Area Efficient Characteristic Computation 118
7.4. Summary 119

Summary and Discussion of Part II 121

Part III – Infrastructure Reuse for Offline Testing 123

8. Test Access through Infrastructure Reuse 125
8.1. Unified Architecture 126
8.2. Test Application 127
8.3. Observing a Test Response 128
8.4. Controlling a Register by Bit-Flipping 129
8.5. Test Access Efficiency 130
8.6. Summary 130

9. Test Sequence Generation 131
9.1. Modeling the Test Sequence Generation 132
9.1.1. Circuit Modeling 133
9.1.2. Fault Modeling 133
9.1.3. Sequential Mapping and Modeling of Bit-Flips 134
9.2. Optimal Test Sequence 135
9.3. Bit-Flipping Scan Test Sequence Generation 136
9.4. Summary 139

10. Experimental Evaluation of the Offline Test Scheme 141
10.1. Experimental Setup 141
10.2. Area Overhead 143
10.2.1. Dependence on Register Size 143
10.2.2. Application to Benchmark Circuits 144
10.3. Test Application Time 146
10.4. Test Data Volume 147
10.5. Peak and Average Test Power 148
10.6. Test Energy 150
10.7. Summary 151

Summary and Discussion of Part III 153

11. Conclusions 155
11.1. Future Research Directions 156

Bibliography 159

Part IV – Appendices 175

A. Tables with Experimental Results 177

A.1. Electronic Design Automation Flow and Tools 177
A.2. Benchmark Circuits 180
A.3. Results - Fault Tolerance Infrastructure 182
A.3.1. Non-Concurrent Detection and Localization of Single Event Upsets 182
A.3.2. Concurrent Online Correction of Single Event Upsets . . . 184
A.3.3. Fault Tolerance in Presence of Multiple Bit Upsets 184
A.3.4. Area Efficient Characteristic Computation 185
A.4. Results - Infrastructure Reuse for Offline Testing 188
A.4.1. Test Access through Infrastructure Reuse 188
A.4.2. Test Sequence Generation 189

Index 197

Curriculum Vitae of the Author 201

Publications of the Author 203

List of Figures

1.1. Bathtub curve depicting the failure rate over time. 3
1.2. Soft Errors: From radiation sources over their effect on semiconductor devices to caused soft errors. 9

2.1. Combinational Circuit C_C. 25
2.2. Graph of C_C. 25
2.3. Sequential Circuit C. 27
2.4. Multiplexer-based Scannable Register (adopted from [BA00]). . . . 35
2.5. Shift Register Latch. 36
2.6. Non-overlapping Clock Scheme. 36

3.1. Architecture of a DRAM with Error Detecting Refreshment (adopted from [HWI+02]). 45
3.2. Principle of Robust Latch Design. 46
3.3. RAZOR Architecture (adopted from [EKD+03]). 48
3.4. GRAAL Architecture (adopted from [Nic07]). 49
3.5. General Embedded Deterministic Test (EDT) Architecture (adopted from [RTK+02]). 52
3.6. General Random Access Scan Architecture. 54
3.7. Multiplexer-based Addressable Random Access Scan Cell. 55

4.1. Presented Non-Concurrent Configurations. 63
4.2. Reference Parity Tree Implementation ($n = 4$). 65
4.3. Schematic of the Parity-Pair Latch (PPL). 66
4.4. Parity Tree Implementation utilizing Parity-Pair Latches (n=4). . . . 67
4.5. Modulo-2 Address Characteristic. 69
4.6. Non-optimal Computation of the Modulo-2 Characteristic. 72
4.7. Optimal Characteristic Tree Organization. 73
4.8. Layout of the Parity-Pair Latch Standard Cell PPL_X1. 76

4.9. Timing Behavior of the OCL Parity-Pair Latch Reference Implementation (DLH_X1 and XOR2_X1): D1-to-Q1 and D1-to-P Delay. 77
4.10. Timing Behavior of the Parity-Pair Latch (PPL_X1): D1-to-Q1, D1-to-P and D2-to-Q2, D2-to-P Delay. 78
4.11. Power and Energy of the OCL Parity-Pair Latch Reference Implementation (DLH_X1 and XOR2_X1) and the Parity-Pair Latch (PPL_X1). 79
4.12. Area Overhead - Parity Computation for a Single Register - Reference Implementation (OCL) and Parity-Pair Latch (PPL). 80
4.13. Area Overhead - SEU Localization at Module Level - Reference Implementation (OCL) and Parity-Pair Latch (PPL). 82

5.1. Presented Concurrent Online Configurations. 87
5.2. Block I and Block II: Deriving and Protecting the Error Condition. . 88
5.3. Block III: Schematic of the Bit-Flipping Latch (BFL). 92
5.4. Timing Behavior in Presence of Soft Errors: a) Unprotected Register (Figure 5.1-a); b) Protected Register with Correction (Figure 5.1-c). . 93
5.5. Area Overhead - Single Error Detection (SED) - Single Register. . . 95
5.6. Layout of the Bit-Flipping Latch Standard Cell BFLATCH_X1. . . . 96
5.7. Timing Behavior of the OCL Low Enable Latch (DLH_X1) and the Bit-Flipping Latch (BFLATCH_X1): D-to-Q Delay. 97
5.8. Power and Energy of the OCL Low Enable Latch (DLH_X1) and the Bit-Flipping Latch (BFLATCH_X1). 98
5.9. Area Overhead - Single Error Correction (SEC) - Single Register. . . 99

6.1. Block I and Block II: Deriving and Protecting the Extended Error Condition in Presence of Double Errors. 106
6.2. Optimal Extended Characteristic Tree Organization. 107
6.3. Area Overhead - Single and Double Error Detection (DED), Single Error Correction Double Error Detection (SECDED) - Single Register. 109

7.1. Detailed Area Overhead Analysis of the Single Error Correction (SEC) Components. 113
7.2. Exclusive OR Truth Tables. 113
7.3. Schematic of the Transmission-Gate Exclusive OR. 114
7.4. Layout of the Transmission-Gate Exclusive OR Standard Cell TGXOR_X1. 116
7.5. Timing Behavior of the OCL Exclusive OR (XOR2_X1) and the Transmission-Gate Exclusive OR (TGXOR2_X1). 117

7.6. Power and Energy of the OCL Exclusive OR (XOR2_X1) and the Transmission-Gate Exclusive OR (TGXOR2_X1). 118
7.7. Detailed Area Overhead Analysis of the Area Efficient Single Error Correction (SEC TG) utilizing the Transmission-Gate XOR. 119
7.8. Area Overhead - Area Efficient Architectures (SED TG, DED TG, SEC TG, SECDED TG) - Single Register. 120

8.1. Unified Architecture. 126
8.2. Bit-Flipping Scan Test Application. 127
8.3. Observing the Compacted Test Response of a Register. 128
8.4. Controlling a Register by Bit-Flipping. 129

9.1. Iterative Bit-Flipping Scan Test Pattern Generation. 132
9.2. Model of a Combinational Circuit ΦC_C (Single Timeframe) and a Target Fault f represented by Φ_f 134
9.3. Sequential Mapping Modeled by Unrolled Timeframes $\Phi C_C, t_{j-1}, \Phi C_C, t_j$ and Model of Bit-Flips $\Phi^B_{t_{j-1},t_j}$ 135

10.1. Schematic of the Bit-Flipping Flip-Flop (BFFF). 142
10.2. Unified Architecture - Area Overhead for a Single Register. 144
10.3. Unified Architecture - Area Overhead for Benchmark Circuits. . . . 145
10.4. Test Application Time for Benchmark Circuits. 146
10.5. Test Data Volume for Benchmark Circuits. 147
10.6. Peak and Average Test Power and Test Energy for Circuit 'b14'. . . 148
10.7. Peak Test Power for Benchmark Circuits. 149
10.8. Average Test Power for Benchmark Circuits. 150
10.9. Test Energy for Benchmark Circuits. 151

A.1. Experimental Setup with EDA Tool and Data Flow. 178

List of Tables

5.1. Time Vulnerability: 8-bit Register with Single Error Correction (SEC). 100

6.1. Double Error Locations - Single Error Correction (SEC, dotted part, Section 5.2) and Single Error Correction, Double Error Detection (SECDED). 105

A.1. Used Electronic Design Automation Tools and Versions. 179
A.2. Properties of the used Public Benchmark Circuits. 180
A.3. Properties of the used Industrial Benchmark Circuits. 181
A.4. Area Overhead - Parity Computation for a Single Register - Reference Implementation (OCL) and Parity-Pair Latch (PPL). 182
A.5. Area Overhead - SEU Localization across Multiple Registers - Reference Implementation (OCL) and Parity-Pair Latch (PPL). 183
A.6. Area Overhead - Single Error Detection (SED). 184
A.7. Area Overhead - Single Error Correction (SEC). 185
A.8. Area Overhead - Single and Double Error Detection (DED), Single Error Correction Double Error Detection (SECDED) - Single Register. 185
A.9. Detailed Area Overhead Analysis of Single Error Correction (SEC) Components. 186
A.10. Detailed Area Overhead Analysis of the Area Efficient Single Error Correction (SEC TG) utilizing the Transmission-Gate XOR. 186
A.11. Area Overhead - Area Efficient Error Detection (SED TG, DED TG). 187
A.12. Area Overhead - Area Efficient Error Correction (SEC TG, SECDED TG). 187
A.13. Unified Architecture - Area Overhead for a Single Register. 188
A.14. Unified Architecture - Area Overhead for Public Circuits. 190
A.15. Unified Architecture - Area Overhead for Industrial Circuits. 191
A.16. Test Time (TAT) and Test Volume (TDV) for Public Circuits. 192
A.17. Test Time (TAT) and Test Volume (TDV) for Industrial Circuits. . . 193

A.18. Peak and Average Test Power (TP) and Test Energy (TE) for Public Circuits. 194
A.19. Peak and Average Test Power (TP) and Test Energy (TE) for Industrial Circuits. 195

Acronyms

ATE	Automatic Test Equipment
ATPG	Automatic Test Pattern Generation
AVF	Architectural Vulnerability Factor
CAD	Computer Aided Design
CMOS	Complementary Metal Oxide Semiconductor
CUT	Circuit Under Test
DED	Double Error Detection
DFT	Design For Testability
DRC	Design Rule Check
DWC	Duplication With Comparison
ECC	Error-Correcting Code
EDA	Electronic Design Automation
EDAC	Error Detection and Correction
FIT	Failures In Time
FSM	Finite State Machine
FT	Fault Tolerance
HDL	Hardware Description Language
IC	Integrated Circuit
LSD	Level-Sensitive Design
LSSD	Level-Sensitive Scan Design
LVS	Layout Versus Schematic
MBU	Multiple Bit Upset
MCU	Multiple Cell Upset
MISR	Multiple Input Shift Register
MTBF	Mean Time Between Failures

MTTF	Mean Time To Failure
NMR	N-Modular Redundancy
OCL	Open Cell Library
PDK	Process Design Kit
PEX	Physical EXtraction
PI/PO	Primary Input / Primary Output
PPI/PPO	Pseudo Primary Input / Pseudo Primary Output
Q_{crit}	Critical Charge
RAS	Random Access Scan
RPG	Random Pattern Generator
RTL	Register Transfer Level
SAF	Stuck-At Fault
SAT	Boolean Satisfiability
SBU	Single Bit Upset
SEC	Single Error Correction
SECDED	Single Error Correction, Double Error Detection
SED	Single Error Detection
SEE	Single Event Effect
SEFI	Single Event Functional Upset
SEL	Single Event Latchup
SER	Soft Error Rate
SET	Single Event Transient
SEU	Single Event Upset
SIC	Single Input Change
SPICE	Simulation Program with Integrated Circuit Emphasis
STF/STR	Slow To Fall / Slow To Rise
STUMPS	Self-Testing Using a MISR and Parallel Shift register sequence generator
TF	Transition Fault
TG	Transmission Gate
TMR	Triple Modular Redundancy
TPG	Test Pattern Generation / Generator
TVF	Time Vulnerability Factor
VLSI	Very Large Scale Integration

Notation

▷ Sets

$\emptyset$	empty set
$\mathbb{B}$	set of Boolean values, $\mathbb{B} = \{\text{true}, \text{false}\}$, denoted as $\{0, 1\}$
$\mathbb{N}^+$	set of positive natural numbers
$\mathbb{N}$	set of non-negative natural numbers including 0, $\mathbb{N} \equiv \mathbb{N}^+ \cup \{0\}$
$\mathbb{Z}$	set of integer numbers
$\lvert \cdot \rvert$	cardinality of a set

▷ Set Operators

$\cup$	union
$\cap$	intersection
$\setminus$	difference

▷ Set Relations

$\in$	element
$\subset$	subset
$\supset$	superset
$\equiv$	equivalence

▷ Boolean Operators

$\bar{\cdot}, \neg$	negation
$\wedge$	conjunction
$\vee$	disjunction
$\oplus$	exclusive disjunction
$\Rightarrow$	implication
$\Leftrightarrow$	equivalence

▷ Boolean function $f : \mathbb{B}^n \to \mathbb{B}$, where $n \in \mathbb{N}^+$

▷ Vectors

$\vec{R}$	vector of values, $\vec{R} := \{r_n, r_{n-1}, \ldots, r_1\}$
$\|\vec{R}\|_1$	l_1-norm of a vector, $\|\vec{R}\|_1 := \sum_{i=1}^{n} \|r_i\|$

If not stated otherwise, the least significant bit (lsb) of a binary vector, that is the bit position in a binary number determining whether the number is even or odd, is noted as the right-most bit.

▷ Hamming distance $\Delta_H(\vec{x}, \vec{y})$ between two vectors $\vec{x}, \vec{y}$ is defined as the number of coefficients in which they differ:

$$\Delta_H(\vec{x}, \vec{y}) := \sum_{j=1}^{n} x_j \neq y_j .$$

For binary vectors $\vec{a}$ and $\vec{b} \in \mathbb{B}$ the Hamming distance is equal to the number of ones in $\vec{a} \oplus \vec{b}$:

$$\Delta_H(\vec{a}, \vec{b}) := \|\vec{a} \oplus \vec{b}\|_1 .$$

Summary

The evolution of digital circuits from a few application areas to omnipresence in everyday life has been enabled by the ability to dramatically increase integration density through scaling. However, the continuation of scaling gets more difficult with every generation and poses severe challenges on reliability.

Throughout the manufacturing process the appearance of defects cannot be avoided and further deteriorates with scaling. Hence, the reliability at timepoint zero denoted by the manufacturing yield is not ideal and some defective chips will produce wrong output signals. For this reason, the presence of such *hard faults* needs to be shown prior to delivery during *test* where *automatic test equipment (ATE)* is used to apply a *test set* that covers a predefined set of modeled defects. As some potential defect locations are hard to test using the chips operational interface, additional dedicated *test infrastructure* is included on chip that provides *test access*.

Throughout the operational lifetime reliability is threatened by *soft errors* that originate from interactions of radiation with semiconductor devices and potentially manifest in sequential state corruptions. With further raising soft error rates aggravated by scaling high reliability is maintained by the inclusion of *fault tolerance infrastructure* able to detect, localize and ideally correct soft errors. Thus, the orthogonal combination of two independent infrastructures elevates the area overhead although test support and fault tolerance are never required concurrently.

This work proposes a unified architecture that employs a common infrastructure to provide fault tolerance during operation and test access during test. Similarities between both fields are successfully exploited and traced back to the combination of an efficient sequential state checksum with an effective state update by bit-flipping.

Experiments on public and industrial circuits evaluate the unified architecture in both fields and show an improved area efficiency as well as successful correction during fault tolerance. During test, the results substantiate advantages with respect to test time, test volume, peak and average test power as well as test energy.

Zusammenfassung

Die Fähigkeit die Integrationsdichte mittels Skalierung drastisch zu steigern, hat die Evolution digitaler Schaltungen von ein paar Anwendungsgebieten zur Allgegenwart im täglichen Leben ermöglicht. Eine Fortführung der Skalierung gestaltet sich jedoch von Generation zu Generation schwieriger und stellt darüber hinaus ernste Herausforderungen an die Zuverlässigkeit.

Das Auftreten von Defekten kann während des Herstellungsprozesses nicht verhindert werden und verschlimmert sich unter Skalierung weiter. Die Zuverlässigkeit zum Zeitpunkt null, ausgedrückt durch die Produktionsausbeute, ist somit nicht ideal und einige defekte Chips erzeugen falsche Ausgangssignale. Aus diesem Grund ist es notwendig vorhandene *permanente Fehler (hard faults)* vor der Auslieferung mittels *Test* zu erkennen. Dabei wird eine vorbestimmte Menge von Defekten in einer *Testmenge* modelliert und diese durch *Testautomaten (Automatic Test Equipment, ATE)* auf jeden Chip angewendet. Da einige der potentiellen Defektstellen mittels der funktionalen Chipschnittstellen nur schwer zu testen sind, wird dem Chip zusätzlich dedizierte *Testinfrastruktur* hinzugefügt, die einen *Testzugriff* bietet.

Während des Systembetriebs wird die Zuverlässigkeit durch *transiente Fehler (soft errors)* bedroht, die durch die Interaktion von Strahlung mit den Halbleitermaterialien hervorgerufen werden. Diese manifestieren sich möglicherweise in Veränderungen des sequentiellen Schaltungszustands. Mit weiter steigenden transienten Fehlerraten, die durch Skalierung verstärkt werden, wird eine hohe Zuverlässigkeit durch das Hinzufügen von *Fehlertoleranzinfrastruktur* beibehalten, die transiente Fehler erkennen, lokalisieren und idealerweise korrigieren kann. Folglich erhöht die orthogonale Kombination zweier unabhängiger Infrastrukturen den Flächenbedarf, obwohl Testunterstützung und Fehlertoleranz nie gleichzeitig benötigt werden.

Diese Arbeit stellt eine vereinheitlichte Architektur vor, die eine gemeinsame Infrastruktur verwendet, um Fehlertoleranz während des Betriebs und Testzugriff während

des Tests bereitzustellen. Ähnlichkeiten zwischen beiden Gebieten werden erfolgreich ausgenutzt und auf die Kombination einer effizienten Zustandsprüfsumme mit einer effektiven Zustandsaktualisierung durch Bit-Flipping zurückgeführt.

Die durchgeführten Experimente für öffentlich verfügbare und industrielle Schaltungen beurteilen die vereinheitlichte Architektur in beiden Gebieten und zeigen eine verbesserte Flächeneffizienz, sowie eine erfolgreiche Korrektur während der Fehlertoleranz. Für die Testunterstützung belegen die Ergebnisse Vorteile in Bezug auf Testzeit, Testdatenumfang, maximale und durchschnittliche Verlustleistung im Testbetrieb sowie Testenergie.

Chapter 1
Introduction

Since the invention of the transistor in 1947, the development and manufacturing of *integrated circuits* (ICs) has undergone a steep evolution as predicted by Gordon E. Moore [Moo65; Moo75]. The ability to scale transistor sizes and increase the integration density of ICs in an economic way constitutes the main driver of this digital revolution. Finally it led to a ubiquitous presence of digital circuits influencing a wide variety of areas and applications, ranging from computers to everyday commodities. In the early days, computers were only affordable for a small number of specialized tasks and were built using discrete ICs containing only a few relatively large transistors. For example, the Apollo Guidance Computer used during the moon landing in 1969 contained 17280 transistors, operated at a frequency of 1 MHz and required 77 W of electrical power. Today (2015), even mobile phones include embedded processors, such as the Samsung Exynos 7 Octa 7420, that are manufactured with feature sizes as small as 14 nm, comprise more than 1 billion transistors and run at 2 GHz while consuming less than 5 W of power.

Over the last decades, the exponentially increasing transistor counts provided by manufacturing technology were used to raise the *performance* of integrated circuits. With the end of scaling in sight, its continuation is faced with growing challenges and the speed of further scaling is prognosed to decline [ITR13]. Historically, some of the difficulties arising from shrinking geometries, such as reaching and maintaining acceptable *yield* during production, could be compensated by sole adjustment and improvement of the manufacturing process. While moving from process generation to generation, other metrics, such as peak and average *power* consumption, stopped scaling proportionally to the transistor dimensions. Hence, the usage and management of power is required to be concerned concurrently to operation throughout the lifetime [Mud01].

However, technology scaling effects have an adverse impact on lifetime *reliability* [SABR04]. It denotes the probability that a device will perform its intended function under stated conditions for a specified period of time [ALRL04]. Reliability has the potential of being the next principal metric being used for integrated circuits [Muk08]. Whenever an IC produces a wrong output signal, called *error* [BA00], it fails to fulfill its specified function. *Failure* of integrated circuits is caused by a variety of mechanisms, which result in *hard faults* and *soft errors*.

Hard faults are permanent and unrecoverable. They relate to the manufacturing process, where *defects*, unintended differences between the implemented hardware and its intended design [BA00], are unavoidable. As a result some ICs are expected to fail. Therefore, *test* is a necessity to prove the absence of hard faults and every manufactured IC undergoes testing in order to assess product quality and quantify the production yield [WWW06].

Soft errors are transient events that reduce the reliability of ICs throughout the lifetime. The continued scaling leads to a dramatic increase in the sensitivity to radiation [Bau08]. Radiation effects in semiconductor devices are responsible for a plethora of reliability issues [Bau08]. These *single-event effects (SEE)* are produced by several types of energetic particles present in the terrestrial environment [Bau08]. The particles travel through the silicon of the device and a part of the particle's energy is transferred to the device. Finally, the deposited energy will result in signal or state corruption [Bau08]. In contrast to hard faults, soft errors cause incorrect operation of ICs without the presence of defects. Consequently, recovery from soft errors is possible.

The likelihood of soft error occurrence mainly depends on two factors: The radiation level and the susceptibility of semiconductor devices to soft errors. The Apollo computer was exposed to high radiation levels during the moon landing. But soft errors were unlikely to occur during the mission time due to the large feature sizes being used resulting in a low susceptibility. In contrast, the feature sizes used in modern ICs, such as the Exynos processor, entail a much higher susceptibility to soft errors, even if operated in environments with lower radiation levels such as sea level on earth. For current transistor technologies soft errors are already an issue [Bau05]. For prospective technology nodes with estimated feature sizes below 6 nm by 2028 [ITR13] integrated circuits are required to employ reliability mechanisms to sustain Moore's prediction, even in domains in which soft errors were not a problem until now [HNG+13].

The remainder of this chapter starts with a discussion of the failure mechanisms responsible for hard faults and soft errors in *Complementary Metal Oxide Semiconductor (CMOS)* materials and transistors. It is followed by a description of common practices in test and design-for-test. Then, the mitigation of soft errors at different abstraction levels is depicted. This chapter closes with the challenges in fault tolerance and test tackled in this work which are followed by the outline of the remaining chapters.

1.1. Failure Mechanisms in Integrated Circuits

Throughout their lifetime, the reliability of integrated circuits is affected by errors that result in a failure of the IC. The classic "bathtub curve" shown in Figure 1.1 is used to depict the *failure rate* over the lifetime.

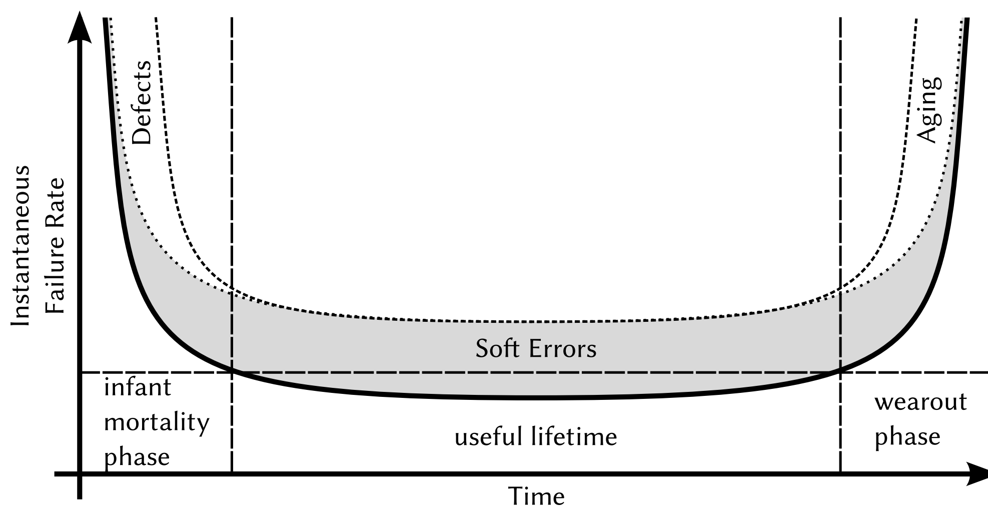

Figure 1.1.: Bathtub curve depicting the failure rate over time for a present technology (solid curve) and a scaled technology (dashed curve).

For any technology three different causes of failure can be distinguished (solid curve in Figure 1.1). During the initial operation of an IC, the failure rate is high. Early-life failures cause newly manufactured hardware to fail and can be attributed to manufacturing problems. After this infant mortality phase, the IC typically works properly with a relatively low constant failure rate until it reaches the end of its useful lifetime. Then, the wearout accelerates and results in significantly higher failure rates caused by a degradation of component characteristics.

Technology scaling impacts the failure rate in all three phases (dashed curve in Figure 1.1). ICs are faced with an increased amount of manufacturing defects and accelerated aging. As a result, the useful lifetime is shortened. During this period reliable operation is further exacerbated by elevated soft error rates.

The following two subsections depict the basic failure mechanisms to provide a broad understanding of the present reliability problems and to accentuate the necessity of test and fault tolerance for prospective technology nodes.

1.1.1. Permanent Faults in CMOS Materials

Permanent or hard faults are caused by two fundamental failure mechanisms: *Extrinsic mechanism* that result in decreasing failure rates and affect a small fraction of the produced ICs and *intrinsic mechanisms* that show increasing failure rates while affecting a large fraction of the manufactured ICs [JED03].

1.1.1.1. Extrinsic Failure Mechanisms

Extrinsic mechanisms are related to manufacturing defects introduced during the production process. They result in early life failure or infant mortality where the failure rate is high directly after production and decreases over time (Figure 1.1). The manifestation of extrinsic errors can be accelerated to identify weak ICs that would otherwise fail early in the field. A process called burn-in is used directly after production that tests the ICs at elevated temperature and voltages.

The typical causes for defects can be attributed to *random* and *systematic* effects. Random effects are related to contaminations and impurities introduced during the production process. Systematic effects relate to the involved process steps such as lithography or polishing. In the following, extrinsic failure mechanisms are summarized according to the book of Chiang and Kawa [CK07].

Random Effects *Impurities* are inevitable during manufacturing and often inherent to the involved processing steps. The most common impurities originate from waver material or chemicals and include dust particles on mask or projection. The likelihood of features being affected increases with every scaling step if impurities are assumed to be of constant size. *Random Dopant Fluctuation (RDF)* describes the random process involved in the implantation of dopant atoms during transistor fabrication. With

dopant count and position not being repeatable and transistor channels containing only tens or few hundreds of dopant atoms in newer technology nodes, the addition or deletion of a few dopant atoms leads to large variations in threshold voltage. *Line Edge Roughness (LER)* denotes variations in the width of features being caused by statistically fluctuating effects inherent to lithography and etching steps, such as photon flux variations, distribution of chemical species in the resist and acid diffusion. Additional details on random effects are found in [BC08].

Systematic Effects *Photolithography* describes the process of exposing photo resist during manufacturing. The latest process nodes use light wavelengths that are larger than the feature sizes to be produced. Difficulties arise from the used masks, where techniques such as phase-shift masks and optical proximity correction are required to approximate desired feature geometries as well as the series of needed lithography steps being affected by effects like depth of focus or misalignment. *Chemical Mechanical Polishing (CMP)* is used to planarize the waver surface between process steps with chemical and mechanical forces. As the surface topography changes across the die with metal density, the material removal leads to wire density dependent erosion and wire width dependent dishing of metal wires.

1.1.1.2. Intrinsic Failure Mechanisms

Intrinsic mechanisms are related to the wearout of materials used in CMOS transistors, such as metal or silicon dioxide. In latest technology nodes, the power supply levels and electric field strengths are already saturated while the clock frequency cannot be raised any further. Thus, continued performance improvements have to be achieved through parallelism at the cost of additional area. Technology scaling is a viable option to confine cost, but goes along with oxide layers consisting of only a few atomic layers, higher chip temperatures, and increased power densities, effects that in turn accelerate degradation mechanisms [HNG+13]. Their failure rate corresponds to the wearout phase in Figure 1.1. It is low in the beginning of the lifetime and increases with time, the IC is affected by *aging*. A detailed discussion of intrinsic failure mechanisms is provided in the book of Segura and Hawkins [SH04].

Thermal Cycling or metal stress voiding is linked to differences in the thermal expansion coefficients of metal and its surrounding isolation. At high temperatures metal expands and tightly bonds to the isolation. At lower temperatures tensile stress arises from thermal expansion and can pull the metal line apart. *Electromigration* describes

the movement of metal under the influence of electron flow and temperature. It affects metal lines if sufficient current density is applied in combination with high temperatures. The metal atoms move and can form voids or extrusions which lead to an open defect or bridge defect. *Time Dependent Dielectric Breakdown (TDDB)* denotes the breakdown of the thin thermally grown silicon oxide used as a dielectric in transistors. Scaled transistors are operated close to their specified voltage with a low electric field strength, that results in an electron tunneling current. When the current is applied for a long time period, a conducting path through the gate oxide to substrate is formed. *Hot Carrier Injection (HCI)* causes damage in NMOS transistors as hot electrons enter the depletion region whenever a transistor is stressed by switching. As a result, the threshold voltage of the NMOS transistor increases and leads to operating frequency reduction. *Negative Bias Thermal Instability (NBTI)* causes threshold voltage shift in PMOS transistors. Transistors are stressed whenever the input is 0 (negative bias) and partly recover during relaxation phases when the input is 1. While the physical cause is not fully understood, the applied electric field is believed to release hydrogen from the oxide/substrate border that results in hole trapping in the oxide. [SH04]

1.1.2. Radiation-induced Soft Errors in CMOS Transistors

Transient events that result in errors without the presence of physical defects can be attributed to environmental effects. Representative failure mechanisms include dynamic variations of the operating conditions (e.g. voltage and temperature) or the interaction of radiation with CMOS materials. The work at hand focuses on radiation-induced soft errors as the decrease of feature sizes and the reduction of operating voltages in order to satisfy the demand for higher integration density, increased functionality and performance as well as reduced power consumption lead to a dramatic increase of sensitivity to radiation [Bau08]. In advanced ICs soft errors have become an important responsibility, as the failure rate attributed to soft errors exceeds the collective failure rate of all other reliability mechanisms [Bau08].

This section depicts the failure mechanisms leading to soft errors according to the book chapter by R. Baumann [Bau08]. The discussion of primary and secondary ionizing radiation as the root cause of soft errors is followed by a description of the interaction of radiation with CMOS transistors. Finally, the resulting single event effects are classified according to their impact on the reliability of integrated circuits.

1.1.2.1. Radiation in the Terrestrial Environment

Ionizing radiation has the ability to interact with the materials being used in integrated circuits. Ions originate from different sources with alpha particles and neutrons being the most important ones in the context of soft errors.

Cosmic radiation is a source of natural background radiation that originates in outer space. Throughout its way down to earth the interaction of cosmic radiation with the earths atmosphere results in complex cascades of secondary particles. The predominant particles produced within the cascade are either short-lived (pions and muons) or are attenuated within the atmosphere (protons and electrons), with one exception: Neutrons. The soft error rate experienced by an IC depends on the energy and flux of neutrons. The *neutron flux* reported in the Jedec standard 89A [JED06] is based on measurements performed in [GGR+04], which determined the neutron flux at sea level at New York City to be 13 neutrons per cm^2 and per hour for energies above 10 MeV. The neutron flux is not constant and varies with the solar cycle, location and altitude. The solar cycle has the smallest influence. High sun activity strengthens the magnetic field around earth, thereby increasing its shielding effect against cosmic rays. The difference in flux at sea level is 30 % or 0.3 X between solar maximum and minimum. Location dependency arises from differences in the strength of earths magnetic field and the neutron flux varies across any two terrestrial sites by a factor of 2 X. Altitude can increase the flux by more than two orders of magnitude, e.g. in an airplane at 10 km the neutron flux is increased by 228 X. Neutrons also vary in their kinetic energy and can be differentiated as *high-energy* and *low-energy neutrons.*

High-energy neutrons exhibit energies above 1 MeV. They interact with the nuclei of semiconductor devices such as silicon (^{28}Si) or oxygen (^{16}O) in an elastic or inelastic reaction. In an elastic reaction a part of the energy of the neutron is transferred to the nucleus resulting in an ejection of the nucleus from its position in the material lattice. A high density of electron-hole pairs is produced in the path of the ion until it is finally stopped. If the reaction is inelastic the neutron is absorbed into the nucleus. As a consequence, the nucleus gets instable and fissions, thereby ejecting secondary ions. These ions then generate a high density of charge (Figure 1.2-I).

Low-energy neutrons or *thermal neutrons* are neutrons that have reached an energy state in the order of 0.025 eV, which is similar to the energy of their surroundings. Despite their low energy, they can interact with boron (B) which is used as a p-type dopant and in borophosphate silicon glass layers. Natural boron occurs in two stable

isotopes, ^{10}B and ^{11}B with an abundance of 20 % and 80 %. The ^{10}B isotope is able to capture thermal neutrons resulting in fission that yields secondary high-energy alpha particles (Figure 1.2-I).

Alpha particles are ions that can directly lead to soft errors. They are emitted when the nucleus of an unstable isotope decays to a lower energy state. Such nuclei can be contained close to the transistors if the used packaging material is contaminated with radioactive impurities. These include lead-based isotopes in solder bumps of flip-chip technology, gold used for bonding wires and lid plating, aluminium in ceramic packages, lead-frame alloys and interconnect metalization. [Bau08]

Modern manufacturing technology employs high purity materials and processes, which results in a significant reduction of alpha particle emission in the finally packaged IC [Nic10]. Processes using aluminium interconnect employ boron precursors which are carefully screened for their ^{10}B content before being introduced to the manufacturing process. Advanced CMOS technologies that employ copper interconnect completely eliminated the use of boron. Hence, thermal neutron induced boron fission is no longer a major source of soft errors [Nic10]. Unfortunately, even latest process technologies are challenged by soft errors as cosmic high-energy neutrons cannot be easily shielded [Muk08].

1.1.2.2. Effect of Radiation on Semiconductor Devices

All previously discussed radiation sources finally produce ions (Figure 1.2-I). In the following, their effect on semiconductor devices is summarized according to [Bau08] which describes the underlying physical background in more detail.

The upper half of Figure 1.2-II depicts a N-channel MOS (NMOS) transistor. If the n^+ node is connected to a positive voltage, a reverse biased n^+/p junction is formed that is especially sensitive to any charge collected from a radiation event.

At the beginning of the radiation event, a cylindrical track of electron-hole pairs is formed as a consequence of the energetic ion's passage, which typically takes less than 0.1 ps (Figure 1.2-II-a). When the resultant ionization track, that incorporates a very high carrier concentration in a sub-micron radius, is close to or crosses the depletion region, charge carriers are rapidly collected by the electric field. As a result, a large current and voltage transient is induced at the node by *drift charge collection* (Figure 1.2-II-b). This 'prompt' charge collection phase is completed within

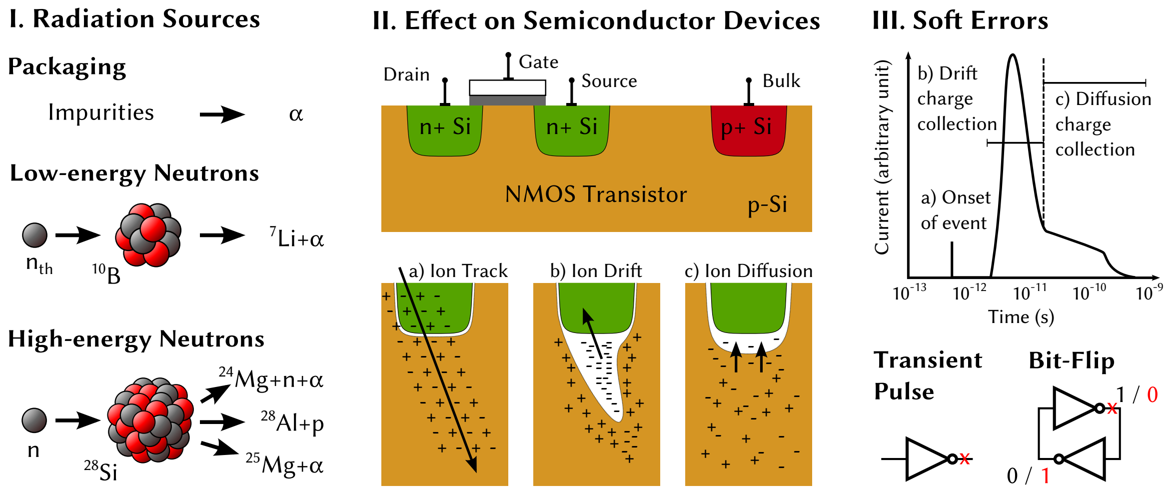

Figure 1.2.: Soft Errors: I) Radiation sources resulting in ions, II) Effect of ions on semiconductor devices: a) Charge generation, b) Drift charge collection, c) Diffusion charge collection - (adopted from [Bau08]), III) Caused soft errors: Transient Pulses (Single Event Transients) and Bit-Flips (Single Event Upsets).

nanoseconds and is followed by a phase of *diffusion charge collection* which lasts hundreds of nanoseconds (Figure 1.2-II-c). The current induced on a single node is depicted over time in the upper part of Figure 1.2-III.

The magnitude of the *collected charge* Q_{coll} depends on the efficiency of the *linear energy transfer* (LET) involved in stopping an ion in matter. Q_{coll} is influenced by a variety of factors either related to the device (such as size, biasing of the various circuit nodes, substrate structure, device doping), the ion (such as type, energy, trajectory) or a combination of both (e.g. the initial position of the event within the device or the state of the device at the onset of the event).

The *critical charge* Q_{crit} denotes the amount of charge needed to change the logic value of a node. It is primarily defined by the node capacitance, the operating voltage and if present, the strength of feedback transistors. Whenever a radiation event strikes a sensitive node and $Q_{coll} > Q_{crit}$ holds, a soft error will be induced. For isolated junctions, such as DRAM cells in storage mode, a simple definition for Q_{crit} suffices:

$$Q_{crit} = C_{node} \cdot V_{node} \quad . \tag{1.1}$$

When storage elements are considered Q_{crit} needs to account for their sequential behavior, thus requiring a broader definition. In latches, flip-flops and static random access memory (SRAM) cells, logic values are stored within an active feedback loop. It is composed of two cross coupled inverters that each consist of a PMOS and a NMOS transistor. Independent of the stored logic value, one of the inverters is driven by a logic 0 and generates a logic 1 at its output. Consequently, its NMOS transistor contains a reverse biased n^+/p junction. If a charged ion traverses the junction, the stored voltage drops due to the collected charge. But as the PMOS transistor is still conducting, it provides a restoring current $I_{restore}$ that recharges the node. The time available for recharging is bound by the switching speed of the storage element, denoted by the time constant τ_{switch}. If the stored value is inverted or not thus depends on whether or not the PMOS transistor contained in the affected inverter can supply enough current to compensate the induced charge before the feedback loop flips to the opposite data state. Hence, Q_{crit} is increased and can be expressed as:

$$Q_{crit} = C_{node} \cdot V_{node} + \tau_{switch} \cdot I_{restore} \quad . \tag{1.2}$$

1.1.2.3. Classification of Soft Errors

The term *Single Event Effect (SEE)* is used to describe all possible effects resulting from the interaction of ionizing radiation with electronic devices [Nic10]. Single event effects comprise physical degradation or breakdown of semiconductor devices that lead to *hard faults* and *soft errors*, which alter the processed data without permanent damage of devices.

Hard faults originating from ionizing radiation are non-recoverable and include mechanisms such as Single Event Latchup (SEL), Single Event Burnout (SEB) and Single Event Gate Rupture (SEGR). Upon occurrence, they cannot be distinguished from hard faults caused by early-life failure or wearout mechanisms. Therefore, hard faults caused by radiation are not explicitly treated in this work, a detailed discussion is found in [Sex03].

Soft errors are recoverable events caused by charge collection in junctions due to the ionizing radiation. Dependent on the position, either the combinational network or the sequential state of a circuit is affected (Figure 1.2-III).

In combinational logic, the term *Single Event Transient (SET)* is used to denote the generation of transient pulses at device nodes. SETs affect semiconductor devices and are visible at the outputs of logic gates. SETs propagate along sensitized paths of a circuit and have a high likelihood of being masked by several effects [SKK+02].

- ▷ *Logical masking* happens when the transient pulse cannot propagate from its origin to a latch due to the lack of a sensitized path.
- ▷ *Electrical masking* occurs as the electrical properties of the passed gates attenuate pulses with insufficient strength or duration before a latch is reached.
- ▷ *Temporal (or latching window) masking* takes place if the pulse indeed reaches a latch but does not satisfy its setup and hold time conditions.

Thus, not all transient pulses caused by radiation events finally result in a soft error.

Single events that directly affect sequential memory elements are called *Single Event Upsets (SEU)*. They directly induce enough charge into the storage structure to reverse or flip the logic value of one or more memory cells, registers, latches or flip-flops [Bau08]. SEUs are further distinguished according to the amount of affected bits and their distribution across memory words or registers. For a *Single Bit Upset (SBU)* only a single bit is flipped. If more bits are inverted in a register or word by a single radiation event, the register experiences a *Multiple Bit Upset (MBU)*. If bits across different registers or words are changed, a *Multiple Cell Upset (MCU)* happened.

The contribution of single event transients and single event upsets to the soft error rate was estimated by Mitra et al. in 2005 [MSZ+05]. They concluded, that SETs affecting static combinational logic account for 11 % of the SER. This is considerably smaller than the 89 % SER contribution of SEUs, that arises from unprotected SRAMs (40 %) and sequential elements (49 %) [MSZ+05]. In 2009, Gill et al. from Intel showed for a 32 nm technology, that the chip-level SER contribution of combinational logic is below 30 % of the chip-level nominal latch SER and hence is not a dominant contributor to the overall SER [GSZ09]. Soft error concerns continue to be exacerbated with scaling [THL+14]. Details on the implications of scaling effects and emerging devices for soft errors are provided in the book of Autran and Munteanu [AM15]. Even under reducing soft error rates per bit, the bit count raises with every technology generation and thus the soft error rate at system level will continue to increase. Soft errors have the potential to dominate the failure rate, whereas SEUs constitute the largest contribution to the soft error rate. Hence, the protection of dedicated memories and sequential elements is the most promising candidate for reliability improvement.

1.2. Test and Design for Test

Test since ever is an essential task in the production process of digital circuits. Test is an experiment to prove the presence of hard faults arising from the production process and is used to assess the quality of delivered ICs. This section depicts the basic concepts and the challenges associated with testing, whereas the book of Bushnell and Agrawal provides a more detailed discussion [BA00].

Functional testing describes the most obvious form of testing. A digital circuit implementing a Boolean function is provided with input assignments in order to exercise the specified functionality of the circuit. For each input assignment, the correct answer of the circuit is known according to the implemented function. The combination of an input assignment and the intended answer is called a *test pattern*, whereas the aggregation of multiple patterns is called a *test set*. The circuits response, the *test response*, is then compared to the expected answer. If a match is found, the test pattern is said to *pass*, otherwise, the pattern *failed*. Functional testing only accounts for the specified behavior of a circuit and does not consider the implementation. Due to the high number of implemented functions and possible input values some defects might not be detected by functional testing, which are described as *test escapes*. Coverage of a circuit can only be defined according to the tested functionalities (for selected values) and no assertion can be made with respect to the coverage of structural defects.

Structural testing is independent of the implemented functionality and exercises the structural implementation of a circuit. The behavior of defects is abstracted with the help of a *fault model*, with a *fault* being specified by its behavior and the affected *fault location* (usually signals or gates). In the most commonly used *stuck-at fault model*, faults can occur at circuit signals, whereas a signal can be either *stuck-at-0 (SA0)* or *stuck-at-1 (SA1)*. A procedure called *Automatic Test Pattern Generation (ATPG)* is used to generate a test set covering all faults contained in the fault set.

Testing can be conducted as *external test*, where test sets are applied to the circuit with the help of *Automatic Test Equipment (ATE)*, also called *tester*. Or the test patterns are generated on-chip by additional circuitry during *Built-In Self Test (BIST)*. Testing is typically performed in the fab between different production steps and prior to delivery of ICs, which is called *manufacturing test*, or, with the help of BIST infrastructure as an *in-field test*, e.g. during power-up of safety critical devices such as cars.

1.2.1. Testability and Test Infrastructure

Testability is the primary metric used in test. In order to detect a fault, the according test pattern needs to excite the fault location to a desired value, but some locations are harder to excite than others due to the circuit structure or the presence of sequential elements. *Controllability* is defined as the difficulty to drive a signal to a desired logic value [Rut72]. In addition, the logic value of the fault site must be propagated to a circuit output in order to determine if the test pattern passed or failed. *Observability* is defined as the difficulty to observe the logic value of a signal [Gol79]. The first testability metric inheriting both aspects is the 'Scandia Controllability/Observability Analysis Program' [GT80].

Testability can be increased by *design for test* and the introduction of *test infrastructure*. *Scan design* [EW77] is the most widely used test infrastructure to increase the controllability and observability of sequential elements. A *scan chain* is a register composed out of latches or flip-flops. In addition to the parallel access provided to the circuit, it implements an additional test mode during which the scan-chain behaves like a shift-register. The register values can then be read and written over two additional signals in a serial way by bitwise shifting the chain. If all sequential elements of a circuit are added to scan-chains, the circuit is said to be equipped with *full scan*. Although the introduction of scan design is able to significantly increase testability, it incorporates additional area overhead, additional pins for *test access* and increased *test time* as well as *switching activity* for the necessary shift operations.

During test application, the test patterns have to be provided to the *circuit under test (CUT)* and the test responses have to be fetched and checked. The amount of all data exchanged by the ATE and the CUT is denoted by the term *test data volume (TDV)*, or *test volume*. *Test Compression and Compaction* is used in conjunction with scan design to reduce the bandwidth and pin count of the interface between CUT and ATE. Test compression reduces the test volume delivered to the CUT. The test set is compressed losslessly, thereby reducing the test volume and the amount of needed tester pins. Additional infrastructure added at the input side of the CUT in form of a decompressor is then used to regenerate the original test set. Test compaction reduces the test volume of the test responses with additional infrastructure, a compactor at the output side of the CUT. It provides a (potentially) lossy compaction of test responses in the space or time domain (or a combination of both). The compacted test responses are then compared with pre-computed responses in the ATE.

1.2.2. Test Economics

Increasing integration densities and raising functionality due to continued scaling have a considerable impact on *test cost* [ITR13]. Test cost arises from the costs associated with test equipment, the on-chip test infrastructure, and the test application. The cost for testing a single CUT then depends on the required tester capabilities (e.g. pin count, speed), the area occupied in the CUT by added test infrastructure, as well as the test time. In addition to testability, secondary metrics play an important role during test as they directly influence the test cost, and thereby the product cost.

Area Overhead. Test infrastructure is added to increase testability, ease test access and reduce the test time and volume. The area overhead associated with test infrastructure is often considered critical in terms of cost, as test infrastructure is often solely used to facilitate testing, but not used during functional operation.

Test Application Time. During volume production, a high amount of ICs is produced in short time. The amount of testers needed to test all ICs during production is determined by the time needed to test a single IC. Thus, test cost scales nearly linear with test time reduction.

Test Data Volume. The amount of test data exchanged with the CUT during test application defines the ATEs minimum memory configuration. As test cost is coupled to the amount as well as the configuration of the required ATEs, test data volume reduction helps towards using adequate ATE configurations with reasonable cost.

Peak and Average Test Power. During test, the power consumption of ICs can be an order of magnitude higher compared to functional operation due to increased switching activity. The *peak power* determines the dimensioning of the circuits power grid. With a raised peak power during test, either the power grid needs to be reinforced and thus overdimensioned beyond normal operation or test quality might be impacted due to voltage droop. The *average power* is closely related to the thermal design power of an IC. Thus, with a cooling system optimized for functional operation, the heat dissipation during test is limited and elevated temperatures must be compensated by either increasing the test time or by changes in the test architecture.

Test Energy. An average power consumption in excess of the provisioned cooling capabilities can be compensated for short periods of time by exploiting the thermal capacitance of the used materials. Thus, a lowered test energy as the product of test time and average test power enables test conduction under confined cost.

1.3. Soft Error Mitigation

In order to facilitate reliable operation of digital circuits in the presence of soft errors, the soft error rate needs to be confined to a feasible level. A variety of possibilities exists to reduce the probability of transient soft errors. They span across all abstraction layers and will be discussed with respect to their potential in SER reduction. More details are found in [Bau08] and [Muk08].

The occurrence of soft errors can be inhibited by reducing the sources of ionizing radiation or increased shielding. This *source level mitigation* involves the elimination of unstable isotopes and impurities as well as boron or the use of isotopically enriched ^{11}B (see Section 1.1.2.1). In addition, remaining alpha particle emission can be shielded by coating the chip with polyamide prior to packaging or the separation of alpha emitting materials and sensitive circuit components. With all those non-recurring actions already being exploited in current technologies, a further improvement cannot be expected.

Process technology mitigation techniques reduce the collected charge Q_{coll} by process and technology choices. The use of buried implants, which increase substrate doping, reduces the size of the formed funnel and increases substrate charge collection, thereby reducing the charge collected in sensitive nodes. Isolating the well and actively biasing it reduces the charge collected by reverse biased drain nodes at the cost of additional mask and implant layers. Penalties arise in terms of performance (decreased speed due to increased parasitic input capacitance) and area (to accommodate the well). For example, compared to conventional silicon processing (bulk CMOS), the introduction of partially depleted silicon on insulator (SOI) technologies results in a 5 X improvement in SER robustness [RGF+03]. Fully depleted SOI offers a higher SER immunity by eliminating the floating body effect and thus preventing the formation of bipolar junction transistors. Process technology solutions are a limited path for SER reduction, as for a majority the SER reduction is below 100 X at the expense of additional process complexity, yield loss and substrate cost [Bau08].

Design mitigation denotes changes in the design of logic cells. In a SRAM cell, two cross-coupled inverters are employed to store a logic value. The critical charge Q_{crit} to flip the stored logic value is a function of the storage node capacitance, the voltage and the restoring current supplied by the pull-up/pull-down network (see Section 1.1.2.2). The critical charge of such structures can be increased if the restoring current is increased by additional transistors or transistor resizing. Another possibility is to

add resistance between the two inverters in order to increase the time to flip a cell. Thereby providing the pull-up/pull-down transistors with more time to restore the node voltage before a flip occurs but effectively slowing down the SRAM cell. Design mitigation incurs additional area overhead to accommodate larger or additional transistors or additional resistors and is no longer reasonable for clock periods below 1 ns, as adding resistance will constrain the operating frequency.

Hardened latches and flip-flops increase the resilience to soft errors by adopting cell-internal filtering or local redundant design. While their ease of integration seems to be attractive, the soft error rate improvement of hardened elements has to be traded-off against unavoidable impact on area, delay, and power, or a combination thereof [GJD+14]. Moreover, the confinement of hardened elements to a local context renders it difficult to determine their integrity after production, disable the fault tolerance during test, or even obtain the location of detected errors at a higher abstraction level. Hardened sequential elements are immune to ion hits that induce charge on a single internal node. However, in deep-submicron technologies the proximity of circuit nodes within a hardened cell results in charge collection at multiple nodes when a single ion strikes a node [ASW+07]. As further scaling reduces the proximity as well as the critical charge of nodes, the SER improvement that can be expected by hardened sequential elements is limited [GJD+14].

System level redundancy targets soft errors at a high abstraction level by adding redundancy in order to detect, localize and correct soft errors. In its simplest form, a parity bit is computed and stored for each register or memory word. Comparing the parity of the stored data and the stored parity bit allows for the detection of soft errors affecting a single bit (SBU) while the detection of MBUs is not guaranteed. A localization of SBUs within the stored data is not possible and a correction needs to be performed by recomputation. In dedicated memory blocks, the most prominent method to deal with soft errors is the use of information redundancy in form of *error detecting and correcting (EDAC)* codes. By employing multiple check bits, it enables the localization and correction of soft errors. Memories equipped with built in *error correcting code (ECC)* protection are generally available and widely used in many environments and applications demanding for raised reliability levels. Due to the regular organization of dedicated memories, the area overhead is relatively small, as the circuitry added for code computation and consistency checking can be shared across memory words. The reduction in soft failure rate provided by EDAC/ECC protection is significant and typically effective error rates are improved by more than 10,000 X [Bau08].

In summary, soft errors have been shown to dominate the reliability decrease observed with every new process generation. Single event upsets have been identified as the largest contributor to the soft error rate (see Section 1.1.2.3). The general availability and use of effective fault tolerance schemes for dedicated memories is able to reduce the soft error rate. Revisiting the estimates from Section 1.1.2.3, which attributed error rate shares of 11 % due to SETs in combinational logic, 40 % due to SEUs in SRAMs and 49 % due to SEUs in sequential elements, shows that the soft error rate can be reduced by 40 % under the assumption of all dedicated memories (SRAMs) being perfectly protected. If the remaining soft error rate of 60 % is not sufficiently low for reaching a desired reliability level, the protection of sequential elements, with a potential of reducing the remaining soft error rate by over 80 %, is the next logical consequence.

1.4. Overview and Contributions

This chapter identified the following substantial challenges to sustain and advance the reliability of digital circuits:

- ▹ The emerging need for fault tolerance to provide soft error resilience throughout the lifetime, with the protection of sequential elements by means of information redundancy as the next logical challenge.
- ▹ The necessity to prove the presence of hard faults by offline testing, with challenges arising from test economics demanding for a further reduction of test application time, test data volume, test power and test energy.
- ▹ The capability to exploit potential synergies in area overhead if both fields are targeted by a unified architecture.

The remainder of this work is organized in three parts as follows:

Part I - Formal Foundation and Related Work

Chapter 2 - Formal Foundation - provides the formal foundation of test and fault tolerance.

Chapter 3 - Related Work in Soft Error Mitigation and Test Access - reviews widely used test architectures along with their amendments as well as alternative test approaches. Soft error mitigation schemes providing fault tolerance in dedicated memories are reviewed. For random logic, existing solutions implemented at different abstraction levels are discussed with a focus on fault tolerance at the architectural level.

Part II - Fault Tolerance Infrastructure

Chapter 4 - Non-Concurrent Detection and Localization of Single Event Upsets - depicts how information redundancy is employed to detect single event upsets during clock-gated phases. The detection of SEUs is implemented area efficiently by a new standard cell and a localization of failing registers is enabled by a checksum with register granularity. The impact on power consumption is confined by focusing the protection on the clock-gated phase.

Chapter 5 - Concurrent Online Correction of Single Event Upsets - depicts how registers are protected by cross-layer fault tolerance during operation. Applying the checksum computation directly to registers enables the detection of SEUs during operation and their localization within a register with bit granularity. Correction of soft errors is enabled in one additional clock cycle by means of a specialized standard cell allowing to flip the value of an affected bit.

Chapter 6 - Fault Tolerance in Presence of Multiple Bit Upsets - analyzes the effectiveness of the present detection capabilities in presence of double bit upsets. The online architecture is extended in order to detect and distinguish errors with a multiplicity larger than one. The exemplarily performed extension for double errors completely avoids false detections and is implemented area efficiently by merging with the checksum computation.

Chapter 7 - Area Efficient Characteristic Computation - analyzes the area overhead associated with the online architecture. Based on the results, the register checksum derivation is identified as a major contributor to the area overhead which solely utilizes a single standard cell type. The cell is carefully optimized for area and used as a replacement in multiple building blocks of the architecture. The resulting architecture is shown to possess a significantly improved area efficiency.

Part III - Infrastructure Reuse for Offline Testing

Chapter 8 - Test Access through Infrastructure Reuse - depicts how the fault tolerance infrastructure for online correction is reused and extended to provide test access during offline test.

Chapter 9 - Test Sequence Generation - explains how test sequences are generated that fully exploit the capabilities offered by the unified architecture.

Chapter 10 - Experimental Evaluation of the Offline Test Scheme - reviews the application to benchmark circuits. The unified architecture is shown to incorporate a low area overhead due to the integrated consideration of fault tolerance and test. The test generation heuristic is shown to be beneficial in terms of test time, test volume, test power and test energy.

Chapter 11 - *Conclusions* - recapitulates the contributions of this work and indicates future research directions that may benefit from this work.

Part I

Formal Foundation and Related Work

Chapter 2
Formal Foundation

This chapter defines the formal apparatus used throughout this work. First, definitions for combinational and sequential circuits are depicted along with the nomenclature used to describe deviations from the intended design and erratic behavior at different abstraction levels. In the following, the fundamentals of soft errors are described in terms of the used nomenclature, their quantification, and fault tolerance by means of redundancy. In the test domain, the essential fault models are discussed in combination with design for testability by test infrastructure and elementary test algorithms. At last, Boolean satisfiability is introduced as a foundation for Chapter 7.

2.1. Digital Circuits

2.1.1. Modeling Levels

A *digital circuit* is a device that processes input data and produces output data, whereas both the input and the output data are represented by vectors over $\mathbb{B}$. A circuit is well-defined by the size of the input and the output vectors and the mapping between the input and output domain representing the circuit's *function* $f : \mathbb{B}^n \rightarrow \mathbb{B}^m$.

The simplest representation of a circuit is a *truth table* enumerating the resulting output vectors for all possible configurations of the input vector. The applicability of truth tables reduces with growing circuit sizes. By describing the input-output relation of a circuit structural information about the circuit's implementation is absent but might be necessary for some electronic design automation tasks. Hence, alternative ways to model circuits exist in literature and practice. To organize these models different *levels of abstraction (or modeling levels)* can be used while the number of used levels can differ dependent on the intended purpose.

The Y-chart proposed by Gajski and Kuhn and later refined by Walker and Thomas [GK83; WT85] is commonly used to distinguish five abstraction levels that comprise details from three domains of description: The behavioral, the structural and the physical domain. During the top-down design process a higher level description of a design is refined and transformed into a lower level description. Starting from the design specification at the *architectural or system level*, the *algorithmic level* describes the function of a circuit in a *hardware description language (HDL)* without any assumptions on the implementation or internal organization. The *functional block or register transfer level (RTL)* adds structural information by distinguishing sequential and combinational logic. Individual registers are modeled together with constructs familiar from programming languages describing the data and control flow. A *logical (gate) level* implementation is then synthesized from the RTL description. This *gate level netlist* models a circuit as a set of components and a set of signals connecting them. Each component is a (logic) gate and implements a Boolean function. Similar to the composition of functional blocks from gates and connections between them, at the *circuit (transistor) level* individual gates are described in the structural domain as netlists that contain transistors, resistors and capacitances. On the contrary, in the physical domain the actual layout and routing information of individual gates is modeled by the geometric description of the masks used for production.

Throughout this work logic level modeling is assumed due to the following reasons. The abstraction level of a gate netlist provides structural information by modeling single gates and their interconnect, which is necessary to model structural faults, perform fault simulation and to reason about test pattern generation. A logic level gate netlist is also technology-independent. Gates, as the smallest units, are defined by their input and output vectors as well as their Boolean function independent of their physical implementation. If a model at the circuit level is mandatory to assess the area overhead or determine the probability of radiation induced soft errors, a gate netlist can be easily mapped to any technology that provides a (technology-dependent) standard cell library.

2.1.2. Combinational Circuit

A digital *combinational circuit* C_C is a device with n inputs and m outputs implementing a Boolean function $\varphi_{C_C} : \mathbb{B}^n \to \mathbb{B}^m$. In Figure 2.1, the gate $g2$ is called a *predecessor* of $g4$, while $g6$ is a *successor* of $g4$. More general, for any circuit element

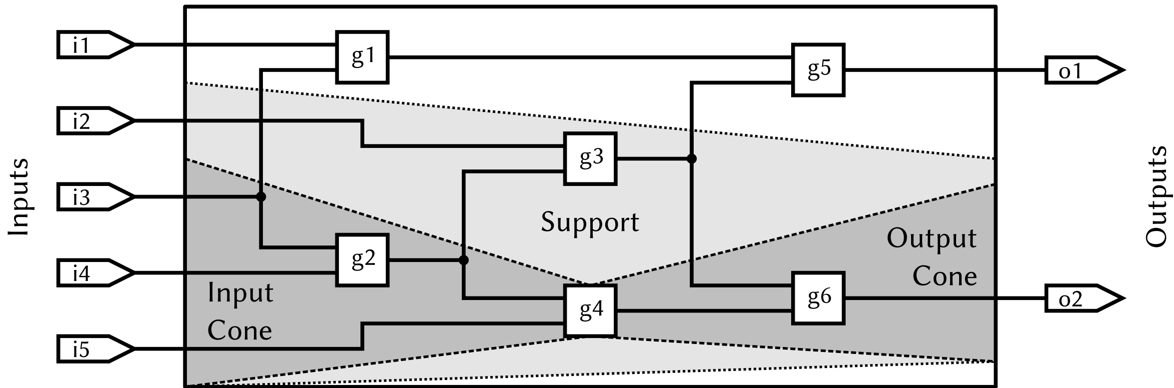

Figure 2.1.: Combinational Circuit C_C.

e the term *input cone* of e denotes the subcircuit containing all predecessors of e, and the *output cone* of e consists of all its successors. The subcircuit denoted as the *support* of e contains all input cones of the outputs that are successors of e.

At the gate level the combinational circuit C_C is represented by a directed acyclic graph (Figure 2.2) which is called a gate level netlist and defined as follows.

Definition 2.1.1 (Combinational Gate Level Circuit) *A combinational gate level circuit C_C is a directed acyclic graph with vertices V and edges $E \subset V \times V$. $V := I \cup G_C \cup O \cup F$ is a disjoint union of the input vertices I, combinational vertices G_C and output vertices O as well as fanout vertices F.*

The edges represent connections between nodes called *nets, wires* or *signals*. The number of connections of a vertex depends on its type. Input vertices have only outgoing edges, while output vertices have exactly one incoming edge. *Fanout vertices* with exactly one incoming edge, the *fanout stem*, and at least two outgoing edges, called *fanout branches*, are used to connect multiple signals. The remaining vertices represent combinational logic gates. Each logic gate $g \in G_C$ with l inputs and one output implements a Boolean function $\varphi_g : \mathbb{B}^l \rightarrow \mathbb{B}$ defined by the gate type.

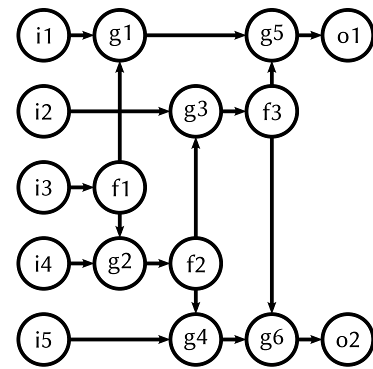

Figure 2.2.: Graph of C_C.

Throughout this work, logic gates with a maximum of two inputs are assumed, as gates with more inputs can be built from two-input gates.

2.1.3. Sequential Circuit

A *sequential circuit* is a circuit whose output function does not solely depend on the values present at its inputs, but on their history (Figure 2.3). This *sequential state* is represented by storage elements, such as latches and flip-flops, which are controlled by a clock signal in *synchronous sequential designs.* Two design styles can be distinguished dependent on the sensitivity to the clock signal. In *level-sensitive designs* latches are used, that are transparent whenever the clock signal has a certain logic value (0 or 1) and latch data by retaining their state during the opposite value of the clock signal. In *edge-triggered designs* that employ flip-flops, new data is latched at a specific clock transition (rising or falling) and stored otherwise.

A *sequential circuit* C with n inputs, m outputs and k sequential elements is a *finite state machine (FSM)* [Mea55]. The up to 2^k states are encoded by the data stored in the sequential elements. The *combinational core* C_C computes two Boolean functions. The *output function* $\varphi_C : \mathbb{B}^n \times \mathbb{B}^k \to \mathbb{B}^m$, that maps pairs of an input and a state to an output and the *transition function* $\tau_C : \mathbb{B}^n \times \mathbb{B}^k \to \mathbb{B}^k$, that maps pairs of an input and a state to the next state. Typically, collections of sequential elements that are accessed together are grouped into *registers* as depicted in Figure 2.3.

2.1.4. Defect, Fault, Error, Failure

This work distinguishes incorrectnesses in digital circuits at different abstractions in accordance to Bushnell and Agrawal [BA00]. At the physical level, the term *defect* is used to describe distortions of the physical shapes in a circuit layout arising from the manufacturing process or during the operation of devices.

Definition 2.1.2 (Defect [BA00]) *A defect in an electronic system is the unintended difference between the implemented hardware and its intended design.*

A *fault* is a formal representation of a defect, that abstracts the physical properties of the infinite and non-discrete range of defects.

Definition 2.1.3 (Fault [BA00]) *A representation of a "defect" at the abstracted function level is called a fault.*

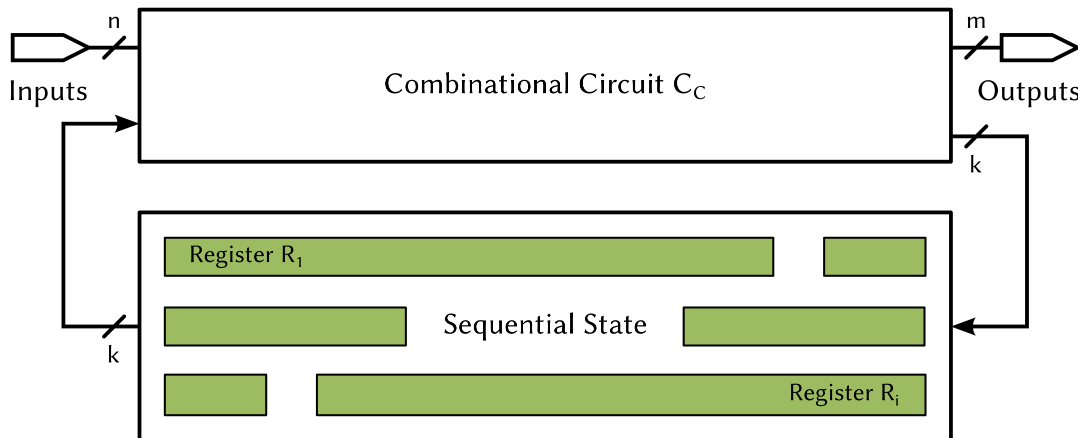

Figure 2.3.: Sequential Circuit C.

If the fault is activated and thus visible at the information theoretical view, it is called an *error.*

Definition 2.1.4 (Error [BA00]) *A wrong output signal produced by a defective system is called an error. An error is an "effect" whose cause is some "defect".*

If an error becomes visible at the system boundary and results in a loss of the intended system function, it is called a *failure.*

Definition 2.1.5 (Failure [Muk08]) *Failure is defined as a system malfunction that causes the system to not meet its correctness, performance, or other guarantees.*

Example Suppose a signal in a circuit that is shorted to the supply voltage signal due to an impurity introduced during the manufacturing process. This defect can be modeled as a fault by the assumption that the signal always has a logic value of 1. It is activated whenever the driving gate produces a logic 0 and manifests as an error as the signal has a logic value of 1 instead of 0. If the error results in a wrong calculation of the circuit, the system fails.

The causes of failure fall into three broad categories [Con03]. *Permanent faults* exhibit a behavior that does not change with time at a fixed location. They are also called *hard faults* and can be attributed to physical defects. In contrast, *non-permanent faults* occur randomly and can be further subdivided by their location. *Intermittent faults*

appear and disappear as a function of time at a fixed location. They relate to marginal or unstable hardware, are activated by environmental conditions, and may evolve into permanent faults. *Transient faults* affect a circuit at random timepoints and random locations. They are caused by environmental conditions such as dynamic parameter variations that lead to violations of timing safety margins [Con03; Bor05] or the charge induced by ionizing radiation [Bau05]. Thus, they are often more precisely denoted as *transient errors* to accentuate the absence of a physical defect or *soft errors* for particle-induced transients.

2.2. Soft Errors

In the following, the foundation of soft errors used throughout this work is established. After familiarizing the nomenclature used to classify soft errors, the basic concepts of soft error quantification are depicted. At last, fault tolerance is discussed with a focus on error detecting and correcting codes.

2.2.1. Used Soft Error Nomenclature

Soft errors have been distinguished and classified in many different ways in literature [Bau08; Muk08; Nic10] and standards [JED06]. The nomenclature of soft errors relevant in this work will be used as follows.

Definition 2.2.1 (Soft Error [JED06]) *An erroneous output signal from a latch or memory cell that can be corrected by performing one or more normal functions of the device containing the latch or memory cell.*

Most soft errors are caused by a single particle and are therefore denoted as *Single Event Upsets*. Single Event Upsets most commonly affect a single sequential element which is expressed by calling them *Single Bit Upsets*.

Definition 2.2.2 (Single Event Upset (SEU) [JED06]) *A soft error caused by the transient signal induced by a single energetic particle strike.*

Definition 2.2.3 (Single Bit Upset (SBU)) *A single event that induces a single bit in an IC to fail at one time.*

If the energy of a single particle is high enough, it can affect multiple bits. With sequential elements organized into registers or memory words, it is important to distinguish the amount of upsets affecting a single word.

Definition 2.2.4 (Multiple Cell Upset (MCU) [JED06]) *A single event that induces several bits in an IC to fail at one time.*

Definition 2.2.5 (Multiple Bit Upset (MBU) [JED06]) *A multiple cell upset in which two or more error bits occur in the same word.*

2.2.2. Soft Error Quantification

The frequency at which soft errors occur is denoted by the *Soft Error Rate (SER)*. It is measured in units of *Failures in Time (FIT)*, where one FIT is defined as the number of failures per 10^9 device-hours. For memories or sequential elements, the soft error rate is often expressed in FIT/device or FIT/Mbit.

The soft error rate of the system components is additive, hence

$$SER_{circuit} = \sum_{i=0}^{n} (SER^{nominal}_{component\ i} \times AVF_i \times TVF_i) \quad . \tag{2.1}$$

Not all radiation events will finally result in a failure of a digital circuit or system due to *logic derating* and *time derating* [NY03]. The *Architectural Vulnerability Factor (AVF)* comprises electrical and logical masking effects and denotes the probability, that a single event transient affecting a node or device will be observed by the system or user [MWE+03]. The *Time Vulnerability Factor (TVF)* contemplates latch window masking effects of sequential elements and denotes the fraction of time a node or device is susceptible to upsets [ST04].

Reliability denotes the probability that a device will perform its intended function during a specified time under stated conditions [ALRL04]. This probability of survival beyond a specified time (timepoint 0 to timepoint t) is commonly designated by the

term survival function while in the technical domain the term *reliability function* $R(t)$ is widely used. It can be calculated as

$$R(t) = e^{-\lambda t} \tag{2.2}$$

where λ denotes a constant failure rate measured in FIT, such as the soft error rate SER. The term *Mean Time To Failure (MTTF)* is often used to denote reliability and is inversely related to the soft error rate.

$$\textit{MTTF in years} = \frac{10^9}{\lambda \times \textit{24 hours} \times \textit{365 days}} \tag{2.3}$$

2.2.3. Fault Tolerance

Fault tolerance, the ability of a circuit to continue its intended operation in presence of faults, can be achieved by means of redundancy, the provisioning of functional capabilities that would be unnecessary in an error-free environment [Lap85]. Common to most solutions, three goals are essential to cope with soft errors:

- Error Detection: The presence of a soft error is recognized.
- Error Localization: The location of the soft error is revealed.
- Error Correction: The original, error-free data is reconstructed.

The addition of *redundancy* to a circuit in order to increase its robustness in the presence of soft errors can be performed at different domains and abstraction levels. *Temporal redundancy* is achieved by repeating an operation multiple times and comparing the results or by sampling the result of a single execution multiple times. *Structural redundancy* includes diversification and n-modular redundancy, where basic building blocks are replicated structurally and compared. The detection of errors is possible with two replicas in *duplication with comparison*, while majority voting with at least three replicas allows for error correction. *Information redundancy* in form of error detecting and correcting codes will be discussed in more detail in the following.

2.2.3.1. Error Detecting and Correcting Codes

In coding theory, *error detection and correction (EDAC)* are techniques that enable the reliable delivery of digital data over unreliable communication channels. Therefore, redundancy is added to a message in order to check the consistency of the delivered message and to recover corrupted data.

In a *systematic code*, the original data is transmitted in conjunction with a fixed number of additional *check bits* that are derived deterministically from the data bits. In order to detect an error, the same algorithm is applied to the received data bits and the result is compared with the received check bits. A common way of detecting errors is the use of a *parity* bit that denotes whether the number of 1 bits in the data word is even or odd.

Definition 2.2.6 (Even Parity) *Let $\vec{d}$ be a vector with n binary values $[d_n, \ldots, d_1]$, then the even parity bit $p(\vec{d})$ is defined as*

$$p(\vec{d}) = \bigoplus_{a=1}^{n} d_a \quad .$$

The introduction of a parity bit increases the Hamming distance between any two valid code words to 2, thereby allowing the detection of single bit errors (and any odd number of errors) while a correction is not possible.

In general, the number of bit errors that can be detected or corrected is determined by the *minimum Hamming distance* of a code, defined as the minimum Hamming distance between any two valid (fault-free) code words.

Definition 2.2.7 (Minimum Hamming Distance) *Let $\vec{c_1}, \vec{c_2} \in C \subset \mathbb{B}^{n+k}$ be two code words and let $\Delta_H(\vec{c_1}, \vec{c_2})$ denote the Hamming distance between $\vec{c_1}$ and $\vec{c_2}$. Then, the minimum Hamming distance $\Delta_H(C)$ of the code C is defined as*

$$\Delta_H(C) := \min_{\vec{c_1}, \vec{c_2} \in C; \vec{c_1} \neq \vec{c_2}} \Delta_H(\vec{c_1}, \vec{c_2}) \quad .$$

In order to detect errors in a or less bits, the minimum Hamming distance is required to be at least $a + 1$. To correct all errors in b or fewer bits, a minimum Hamming distance of at least $2b + 1$ is required. If errors should be detected in a or less bits and

corrected in b or less bits (where $a \leq b$), the minimum Hamming distance needs to be at least $a + b + 1$ [Muk08].

Error detecting and correcting *Hamming codes* employ multiple check bits to ensure an adequate distance between code words [Ham50]. The minimum number of check bits required for sole *Single Error Correction (SEC)* is given by the Hamming relationship [Ham50]:

$$2^c \geq d + c + 1 \tag{2.4}$$

where d is the number of data bits and c is the number of check bits. To correct a single bit error, the 2^c combinations of the c check bits must be able to localize the error in $d + c$ code bits. In addition, they need to represent the fault free case where no error occurred. Hamming codes are commonly expressed as Hamming (n,d), with $n = d + c$ code bits and d data bits. They are *perfect codes*, as the Hamming relationship is satisfied with a minimum number of check bits. *Extended Hamming codes* that allow *Single Error Correction and Double Error Detection (SECDED)* of errors employ one additional check bit to increase the minimum Hamming distance.

Example Let Hamming(7,4) be the Hamming code defined by the *generator matrix* G and the *check matrix* H with mutually distinct columns.

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.5}$$

To determine the to be transmitted codeword $\vec{X} = (c_1, c_2, d_1, c_3, d_2, d_3, d_4)$, the data vector $\vec{D} = (d_1, d_2, d_3, d_4)$ with value $[1010]^T$ is pre-multiplied by G.

$$\vec{X} = G \cdot \vec{D} = [1011010]^T$$

To check a received codeword $\vec{R}$ it is pre-multiplied by H in order to obtain the *syndrome* $\vec{S}$. For an error free codeword $\vec{R}$ (:=$\vec{X}$), the syndrome $\vec{S}$ is the null vector. For a codeword $\vec{R'}$ with a bit error at bit 5 ($\vec{R'} = [1011110]^T$), the syndrome $\vec{S'}$ is not the null vector and its value indicates which bit has been flipped.

$$\vec{S} = H \cdot \vec{R} = [000]^T \qquad \vec{S'} = H \cdot \vec{R'} = [101]^T$$

For an extended Hamming(8,4) code with an additional parity bit p in the codewords $\vec{X}+ = (c_1, c_2, d_1, c_3, d_2, d_3, d_4, p)$ the generator matrix $G+$ and the check matrix $H+$ are extended as follows.

$$G+ = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad H+ = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.6}$$

2.3. Test of Digital Circuits

This section depicts the basic concepts related to the test of digital circuits. These include the formal description of defects by fault models and test infrastructure providing test access in order to increase testability of circuits. In addition, test algorithms important to generate and grade the quality of test sets are briefly introduced.

2.3.1. Fault Models

Several advantages are associated with the modeling of defects as faults. As the problem is abstracted and thereby simplified, the need to describe complex physical effects is omitted. A single fault model can cover many different kinds of failure mechanisms leading to defects. Fault models allow to grade the quality of a test set and enable the automated generation of structural test patterns.

In order to reflect the faulty logical behavior of defects at the gate-level, *structural fault models* are employed. Faults that affect the temporal behavior are modeled by *delay fault models*, which either model changes in the propagation delay of gates as *gate delay faults* or paths as *path delay faults*. Delay faults can be further subdivided according to their associated delay defect size. *Gross delay faults (GDF)*, that are also called *transition faults (TF)*, exhibit a defect size that exceeds the clock period. For *small delay faults (SDF)*, the delay defect size is smaller than the clock period.

While a variety of fault models exist [Wun10], the structural stuck-at-fault model and the transition delay fault model will be depicted in the following.

2.3.1.1. Stuck-At Fault Model

The most widely used fault model is the *stuck-at fault model* [Eld59; GNR61]. It assumes, that a *stuck-at fault (SAF)* affects a single line or signal of a circuit, where the faulty line is permanently set to either logic 0 (Stuck-At-0 or SA0) or logic 1 (Stuck-At-1 or SA1). A large variety of defects can be represented with this simple model, although it cannot model defects that involve multiple signals or time-dependent defects.

Definition 2.3.1 (Stuck-At Fault) *A stuck-at fault saf in a combinational circuit* $C = (V, E)$ *is a pair* $saf \in E \times \mathbb{B}$*, where the first component denotes the fault location and the second component denotes its polarity.*

2.3.1.2. Transition Delay Fault Model

The transition fault model supposes, that a single gate in a circuit exhibits a delay increase over its nominal value. The delay of the faulty gate is assumed to be sufficiently large to prevent a passing transition from reaching any output within the circuits clock period, independent of the path involved. Thus, transition faults are also called gross-delay faults and either affect the $0 \rightarrow 1$ transition (Slow-To-Rise, STR) or the $1 \rightarrow 0$ transition (Slow-To-Fall, STF) of a gate, usually defined with respect to the transition of the gate output.

Definition 2.3.2 (Transition Fault) *A transition fault tf in a combinational circuit* $C = (V, E)$ *with gates* $G_C \subset V$ *is a pair* $tf \in G_C \times \{STR, STF\}$*, where the first component denotes the affected gate and the second component denotes the polarity of the output transition exhibiting a delay greater than the system clock.*

2.3.2. Test Access through Scan Design

During test, input stimuli are applied to a circuit and the circuits answer is compared to the expected test response. For combinational circuits this is achieved by controlling the (primary) inputs and observing the outputs. In sequential circuits, where an internal state is stored in latches or flip-flops, the testability of these sequential elements is crucial. *Scan design* provides direct controllability and observability of the sequential elements [WA73; EW77]. The original sequential elements are replaced by scannable implementations, that are connected serially in order to form a shift register, called a *scan chain*.

In *edge-triggered designs* where a single clock is used to control the sequential circuit part, a scannable register can be implemented by multiplexed flip-flops as depicted in Figure 2.4. The test control signal *ScanEnable* is used to select if the flip-flop's inputs are connected to the combinational circuit in *system mode*, or to the outputs of the predecessing flip-flops in *test mode*.

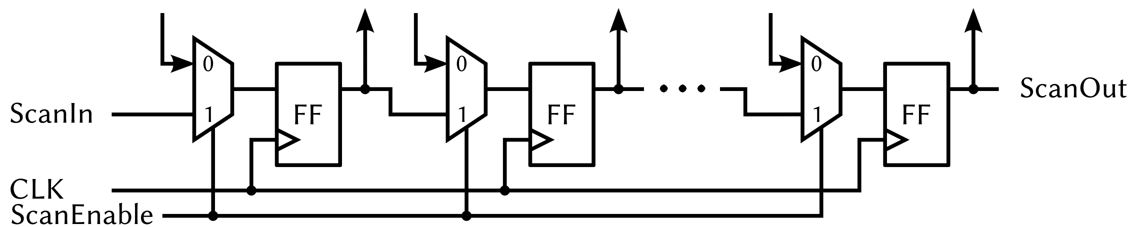

Figure 2.4.: Multiplexer-based Scannable Register (adopted from [BA00]).

In *level-sensitive designs*, that are considered especially robust against timing variations, *level sensitive scan design (LSSD)* is used [EW77]. The $L1/L2$ *shift register latch (SRL)* depicted in Figure 2.5-a is controlled by one system clock and two test clocks A and B. In system mode, the *L1* latch is used to store data from the combinational circuit controlled by clock *Clk*. In test mode, the input *ScanIn* is controlled by the two non-overlapping clock signals *A*,*B*. Hence, the *L2* latch is only used to implement the shift mode. Also in system mode, latches have to be controlled by a non-overlapping clock scheme. The simplest scheme is shown in Figure 2.6. The times τ_1^l and τ_1^h are the low resp. high phases of clk_1, τ_2^l and τ_2^h are these phases of clk_2, and clk_1 and clk_2 are never high at the same time. As these latches are available anyway they can be combined to so called *L1/L2** latches to implement a single SRL (Figure 2.5-b), where the *L2* latch is no longer redundant.

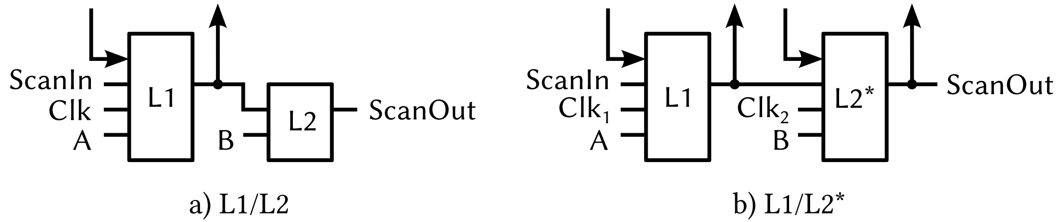

Figure 2.5.: Shift Register Latch.

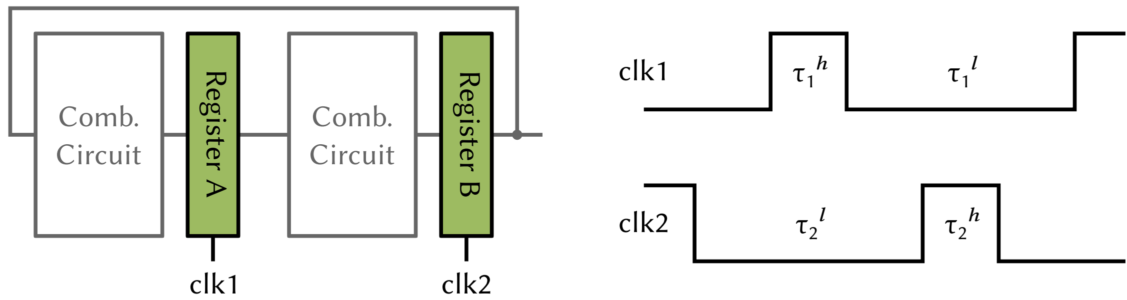

Figure 2.6.: Non-overlapping Clock Scheme.

An overview on the vast amount of different scan architectures targeting different design styles is found in [BA00].

Hence, in presence of scan design, arbitrary combinational test patterns can be applied as a sequential circuit C with n primary inputs (PI), m primary outputs (PO) and k sequential elements behaves like a combinational circuit C_C with $n + k$ inputs and $m + k$ outputs. The additional k in- and outputs are called *pseudo primary* in- and outputs (PPI and PPO). Test application of such a test set is conducted as follows. Each test pattern p_i consists of two parts, a sequential state s_i and an assignment of the primary circuit inputs v_i. The expected response of a pattern similarly contains values of the primary outputs o_i and a sequential state s'_i. The sequential state s_i is shifted serially into the scan chain(s), thereby guaranteeing that the scannable sequential elements are in the desired state. Then, the inputs v_i are assigned and the circuit is operated in normal mode for one clock cycle during the *capture cycle*. Afterwards, the circuit outputs are compared to the expected values o_i and the sequential state is shifted out and compared to s'_i. At the same time, the sequential state s_{i+1} of the next pattern is shifted in.

The time to apply a set of test patterns is called the *test application time (TAT)* or *test time*, which is dominated by the shift operations of the scan chains. The *test data*

volume (TDV) or *test volume* denotes all bits exchanged with the circuit during test. During test application, the peak and average *test power consumption* as well as the *test energy* are elevated due to the shift operations causing a high switching activity in the scan chains as well as the combinational circuit [IZW+07; EWI+08].

2.3.3. Test Algorithms

2.3.3.1. Fault Simulation

Fault simulation denotes the process of simulating a set of test pattern T for a circuit C in presence of faults from a fault set F. For every input stimulus $t \in T$, a logic simulation of the *fault free circuit*, the *good value simulation*, is conducted to determine the fault free circuit response. Then, the circuit model is modified during *fault injection* in order to represent the presence of a fault $f \in F$ and the resulting *faulty circuit* is simulated. If the test response of the good and faulty circuit differ, t is said to be a test for fault f as fault f is controlled and observed under test t.

Typically, the number of test patterns in T and the number of faults in F that need to be considered are large. Fault simulation can be accelerated by *fault dropping*, where faults that have already been detected by previous patterns are removed from the fault list. Or *parallel-pattern single-fault propagation (PPSFP)* is employed, where multiple patterns are evaluated concurrently during the simulation of a single fault [WEF+85].

During fault simulation of a test set T, the faults of a fault set F are distinguished into *detected* and *undetected* faults, where $F = F_{detected} \cup F_{undetected}$. This classification is used as a metric for the quality of a test set T, the *fault coverage*.

Definition 2.3.3 (Fault Coverage) *Given a circuit C, a set of faults F and a test set T, then the fault coverage $FC(T)$ is defined as*

$$FC(T) := \frac{|F_{detected}|}{|F|} \times 100\,\% \quad .$$

A perfect fault coverage cannot always be reached as some of the faults in a circuit are *undetectable* ($F = F_{undetectable} \cup F_{detectable}$ with $F_{detectable} = F_{detected} \cup F_{undetected}$). Hence, the *effective fault coverage* or *fault efficiency* is used as a metric, where 100 % denote the maximum quality that a test set can reach.

Definition 2.3.4 (Fault Efficiency) *Given a circuit C, a set of faults F, a set of proven undetectable faults $F_{undetectable} \subset F$ and a test set T, then the fault efficiency $FE(T)$ is defined as*

$$FE(T) := \frac{|F_{detected}|}{|F| - |F_{undetectable}|} \times 100\,\% \quad .$$

2.3.3.2. Test Pattern Generation

In case of an insufficient test coverage, additional test patterns are required to target yet undetected faults. The process of generating test patterns that show the presence of faults in a circuit is called *Automatic Test Pattern Generation (ATPG)*, which is known to be a NP-complete problem [IS75]. A brief introduction to test generation is given here, while more details are provided in literature [BA00].

The ATPG process needs to fulfill two conditions to generate a test for a targeted fault: *Fault activation* and *fault propagation*. For stuck-at faults, fault activation requires that the fault location is excited to the opposite logic value of the fault polarity, that is 0 for a SA1 and 1 for a SA0. Fault propagation requires that a sensitized path from the fault location to a primary output exists, over which the fault effect can be observed. For transition faults, pattern pairs $P = (p_1, p_2)$ are required, where the *initialization pattern* p_1 excites the fault location to the initial value of the targeted transition (0 for STR and 1 for STF). The *propagation pattern* p_2 excites the fault location to the final value of the targeted transition (1 for STR and 0 for STF), and sensitizes a path from the fault location to a primary output.

Test generation can be conducted by traversing the structural circuit description in *structural ATPG*, or can be formulated as a Boolean satisfiability problem (SAT) [DEFT09; Czu13]. To satisfy a fault's activation and propagation condition, the input assignment of the test pattern needs to guarantee the required values of circuit signal lines. Typically, this *line justification* only needs to specify a small amount of input bits to derive a test pattern for a single fault. The specified bits of a test pattern are called *care bits*, whereas the unspecified bits are denoted as *don't care* bits. Specifying the don't care bits of a partially specified pattern allows the detection of additional faults, which can be determined by fault simulation of the filled pattern. Limiting the maximum number of specified bits increases the efficiency of test compression and compaction in reducing the test time and volume [KZIW08].

2.4. Boolean Satisfiability

Boolean Satisfiability (SAT) is the problem of deciding whether a given propositional formula can be satisfied. For a Boolean formula $f : \mathbb{B}^n \rightarrow \mathbb{B}$, satisfiability determines if an *interpretation*, an assignment $a \in \mathbb{B}^n$ of the variables, exists for which f evaluates to *true*: $f(a) = 1$. Any Boolean formula can be represented in *conjunctive normal form (CNF)* defined as follows:

Definition 2.4.1 (Conjunctive Normal Form) *A conjunction of clauses* $\bigwedge_i c_i$*, each* clause c_i *being a disjunction of literals* $\bigvee_j l_{i,j}$*, and each* literal $l_{i,j}$ *being either a Boolean variable* v *or its negation* $\neg v$*.*

Example

$$CNF(f) = (l_{1,1} \vee l_{1,2} \vee l_{1,3}) \wedge (l_{2,1} \vee l_{2,2} \vee l_{2,3}) \wedge (l_{3,1}) \wedge (l_{4,1} \vee l_{4,2}) \tag{2.7}$$

The Boolean formula is called the SAT *instance*, whereas the interpretation of the instance variables is called the SAT *model* or assignment. An assignment of variables for which the formula evaluates to *true* is called a *satisfying assignment*. If such an assignment exists, the SAT instance is said to be *satisfiable* (denoted by *SAT*). If it can be proven, that such an assignment does not exist, the function represented by the formula is immediately *false* for all possible variable assignments and the SAT instance is said to be *unsatisfiable* (denoted as *UNSAT*). If an algorithm implemented in a satisfiability *solver* is guaranteed to prove either the existence or the lack of a satisfying assignment for any arbitrary instance, it is said to be *complete*.

The Boolean satisfiability problem for instances in CNF is NP-complete and hence, algorithms that solve an arbitrary instance of this problem in polynomial time cannot exist until P=NP [Coo71; Lev73; GJ79]. Nonetheless, Boolean satisfiability has been widely adopted in many areas such as electronic design automation or test generation, as many instances in this domain are nevertheless solvable with manageable runtimes [PCK99]. Therefore, sophisticated solvers are employed, that often rely on a search based procedure known as the DPLL algorithm [DLL62]. An overview of state-of-the-art algorithms is found in [BHvMW09].

Most state-of-the-art algorithms employ *learning* techniques to speedup the search for a satisfying assignment. During the search, additional clauses are derived from conflicting assignments and added to the SAT instance in form of *conflict-driven clauses* in order to prune the search space.

In many application areas, the satisfiability instance is iteratively refined and solved multiple times. One example is test pattern generation, where a (fixed) circuit instance is extended with additional clauses representing faults in order to derive a test set for the circuit. If an instance can be reused for multiple consecutive iterations many time consuming steps such as building an in-memory model or learning in the unchanged instance part have to be executed only once. This is difficult to achieve in presence of learning, as every solver invocation extends the instance with learned clauses, which are in addition only valid for the actual instance. Thus, additional overhead is introduced in order to track and remove outdated learned clauses.

In *incremental solving*, a satisfiability instance $\mathit{CNF}(f)$ is solved under a set of *assumptions* $A = a_1, \ldots, a_n$, where each assumption a_i is a literal that is represented as a unit clause:

$$\mathit{CNF}(f) \wedge a_1 \wedge \cdots \wedge a_n \tag{2.8}$$

Hence, the satisfiability of $\mathit{CNF}(f)$ can be evaluated multiple times under different assumptions by restarting the solver without the need of clause removal [ES03].

In order to evaluate the satisfiability of multiple refined instances a single instance is used, where clauses that refine an original instance are activated conditionally through assumptions. Let $\mathit{CNF}(f)$ be the original instance solved in a first iteration, and let $\mathit{CNF}(f) \wedge (l_{i,1} \vee l_{i,2})$ be a refined instance to be solved in a consecutive iteration. Then, the instance $\mathit{CNF}(f)' := \mathit{CNF}(f) \wedge (a_i \vee l_{i,1} \vee l_{i,2})$ can be used in both iterations by solving $\mathit{CNF}(f)'$ under assumption a_i in the first iteration and under assumption $\neg a_i$ in the second iteration.

The modeling of circuits, faults, and timing behavior as Boolean formulae within SAT-instances will be discussed in the context of satisfiability-based automated test sequence generation in Chapter 9.

Chapter 3
Related Work in Soft Error Mitigation and Test Access

This chapter discusses the related work in soft error mitigation and test access. The protection of regular dedicated memories by information redundancy is followed by solutions that mitigate soft errors in sequential elements embedded in random logic. In order to increase testability, infrastructure for test access, compression and compaction as well as alternative test access architectures are depicted.

3.1. Soft Error Mitigation

Single event upsets affecting memory elements have been identified to be a major contributor to the overall soft error rate (see Chapter 1). This section discusses the related work in protecting dedicated memory blocks as well as sequential elements embedded in random logic.

3.1.1. Dedicated Memories

For dedicated memory blocks information redundancy is employed in form of *error correcting codes (ECC)*. The code computation can be implemented efficiently due to the regular structure and organization of memories. In the following, two solutions will be discussed: *Error correcting codes*, where checksums are computed at memory word granularity, stored in conjunction with the words and checked during read operations. And *error detecting refreshment*, where a signature characterizing the complete memory content is derived, concurrently updated and checked periodically.

3.1.1.1. Error Correcting Code Memory

Nowadays, most server systems provide some form of protection against memory errors. The most commonly used technique is the implementation of *error correcting codes (ECC)*. Thereby, memories, such as ECC DIMMs either provide *single-bit error correction and double-bit error detection (SECDED)*, or use more complex codes for Single Device Data Correction like Chipkill [Del97], that allow to tolerate entire memory chip failures at an elevated energy usage and potentially reduced performance.

Dedicated memories are implemented as a regular array of DRAM-cells. In each cell, a single bit of information is represented by an electrical charge stored in a capacitor. The memory array is addressed row-wise by a row select signal and a single row is called a memory *word*. Thereby a single signal line per memory column, called a *bit-line*, is sufficient to read and write the cells. To protect a 64-bit memory word, 7 check bits are required to correct all single bit errors according to the Hamming relationship (see Section 2.2.3.1). In order to distinguish single from double bit errors an additional parity bit is used. Thus, the resulting extended SECDED Hamming code uses 8 check bits per 64-bit word and is denoted as (72,64)-Hamming code.

Upon writing a single word to the memory, an encoder is used to derive the additional check bits, which are then stored along with the memory word. Upon a read request, the memory word and the associated check bits are fetched, decoded and checked for consistency. Hence, in case that a memory cell's content is corrupted by a soft error (called a latent error), the error will not manifest until it is accessed. To cope with the high detection latency if read operations are infrequent, the complete memory content can be read periodically. This *memory scrubbing* also prevents the accumulation of errors into more severe forms that exceed the capabilities of the used code. Scrubbing is performed during idle cycles with a low frequency to limit the memory power consumption and contention. As the encoder and decoder are shared among all memory words, the overhead is dominated by the additional memory cells needed to store the ECC bits, which is 12.5% if 64-bit words are used.

The use of SECDED protection in memories has been reported to decrease the observed soft error rate by 10000 X [Bau08]. In [LSHC07; LHSC10] the SER of unprotected memories is reported to be in a range of thousands of FIT per machine. The application of ECC results in a SER reduction to 1000 FIT, which are attributed to hard faults. The two studies in [SPW09; HSS12] show, that the use of more complex

correction mechanisms, such as Chipkill, is also effective for a significant portion of hard faults as transient errors are almost eliminated by ECC protection.

3.1.1.2. Error Detecting Refreshment of Memories

Soft errors affecting memories can also be addressed by periodic maintenance testing, where a signature of the complete memory content is derived in order to check the memory content for consistency. To calculate such a signature, a counter (or LFSR) is used to enumerate all memory addresses and the memory content is fed to an output data compactor (MISR). Then, the final state of the compactor represents a characteristic of the memory content. This *reference characteristic* C_{ref} is learned and stored during the initialization phase. In order to check the memory for consistency, the same procedure is repeated periodically and the newly derived characteristic C_{test} is compared to C_{ref}. While being able to detect soft errors affecting the memory content, such a scheme is challenged in two ways: As the reference characteristic C_{ref} characterizes the memory content, a new learning phase is required after every write operation. In addition, the explicit enumeration of the memory content to calculate C_{test} involves a high error detection latency.

Both challenges are addressed by the BIST architecture for embedded DRAMs presented in [YHW98; HWI+99; HWI+02]. Assume without loss of generality a bit oriented memory array M containing $m \cdot n$ bits at addresses $A = \{m \cdot n, \ldots, 1\}$. Further, let the set of all addresses where M contains a logic 1 be denoted by $A_1 := \{a \in A | M[a] = 1\}$. Then, the bitwise modulo-2 sum of all binary addresses in A_1 is calculated by

$$C = \bigoplus_{a \in A_1} a \tag{3.1}$$

which characterizes the complete memory content in $l = \lceil \log_2(m \cdot n) \rceil$ characteristic bits and is used as the reference characteristic C_{ref}. Write operations modifying the memory content need to be reflected in this reference characteristic. Therefore, the new characteristic C_{ref}^{new} can be derived from the old characteristic C_{ref}^{old} if the written address is known along with the old and new data at that address. As writing a single bit in embedded memories is only possible by fetching a complete data word to the refreshment register, updating it and writing it back, the reference characteristic is updated concurrent to a write operation by calculating

$$C_{ref}^{new} = C_{ref}^{old} \bigoplus a \cdot (M[a]^{new} \bigoplus M[a]^{old}) \quad . \tag{3.2}$$

As DRAM cells need to be periodically refreshed for data retention, C_{test} is calculated concurrently to their refreshment, thus providing low error detection latencies.

The modulo-2 address characteristic has been shown to incorporate the following properties:

- ▷ All single bit upsets are detectable and $C_{ref} \oplus C_{test}$ provides the address of the affected memory cell.
- ▷ All double bit upsets are detectable and $C_{ref} \oplus C_{test}$ is a sum of two addresses a_1 and a_2, where $a_1 \neq a_2$ implies $C_{ref} \oplus C_{test} \neq 0$.
- ▷ Data compaction based on the modulo-2 address characteristic has been shown to be equivalent to serial signature analysis. Hence, the probability of aliasing is estimated by 2^{-l}, where l denotes the length of the characteristic.

The complete architecture for a row oriented memory array with m rows and n columns and memory addresses split into row and column addresses $a = (a_r, a_c)$ is depicted in Figure 3.1. In order to derive the characteristic C concurrent to the refreshment, the row characteristics C_r are calculated in a single clock cycle from the refreshment register. The global characteristic C is then obtained as

$$C = \bigoplus_{1 \leq a_r \leq m} C_r \tag{3.3}$$

and each row characteristic is defined as

$$C_r = \bigoplus_{a_c \in A_1(r)} (a_r, a_c) = (\bigoplus_{a_c \in A_1(r)} a_r, \bigoplus_{a_c \in A_1(r)} a_c) \tag{3.4}$$

where $A_1(r) := \{a_c | M[a_r, a_c] = 1\}$ denotes the set of all column addresses containing a 1 in row r. The first component of C_r depends on the parity of the row entries and is either a_r or 0. The second component is a modulo two sum with size $\lceil \log_2(n) \rceil$ of all column entries that contain a 1 in row r. It can be implemented by a set of parity trees as

$$\bigoplus_{a_c \in A_1(r)} a_c = (\bigoplus_{a_c \in A_1(r)} a_c^1, \ldots, \bigoplus_{a_c \in A_1(r)} a_c^l) \tag{3.5}$$

where a_c^i denotes the i'th component of $a_c, 1 \leq i \leq \lceil \log_2(n) \rceil$.

While the scheme is able to detect single and double bit upsets in the memory, only single bit upsets are correctly localized. As the modulo-2 characteristic has a

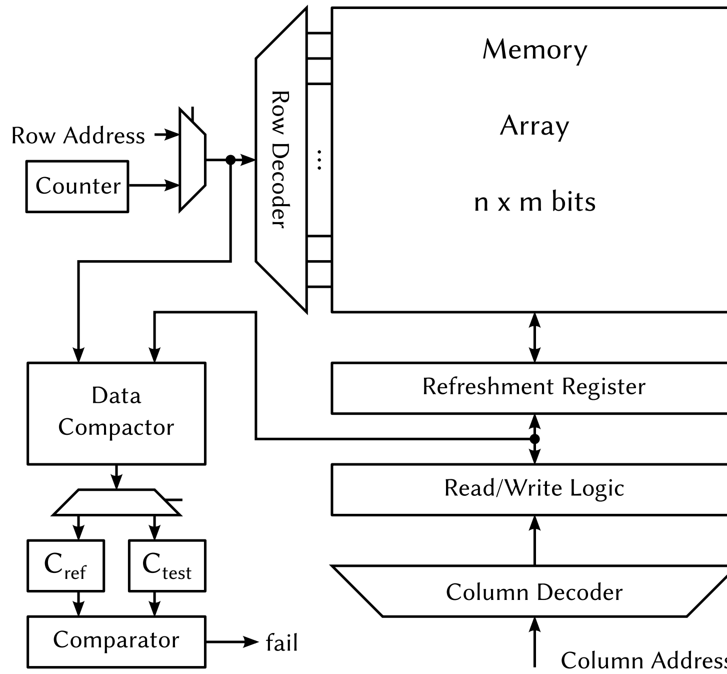

Figure 3.1.: Architecture of a DRAM with Error Detecting Refreshment (adopted from [HWI+02]).

Hamming distance of 3, it cannot distinguish a single from a double bit upset. Hence, a correction can only be attempted if double bit upsets are not assumed to occur.

To distinguish single from double bit upsets, [MIY07] uses a characteristic with an incremented Hamming distance. By appending a constant 1 to every memory address, Single Error Correction Double Error Detection (SECDED) is achieved in analogy to the additional row of 1s in an extended Hamming code check matrix (see Eq. 2.6).

3.1.2. Sequential Elements in Random Logic

With the availability and use of fault tolerance for dedicated memories, the protection of sequential elements embedded in random logic is the next logical consequence in further reducing the overall soft error rate. In contrast to dedicated memories organized in a highly regular fashion additional challenges arise, as sequential elements are distributed throughout the layout of circuits. As discussed in the introduction (Section 1.3) soft errors in sequential elements can be mitigated at different abstraction levels ranging from the process level, over the circuit and architectural level

to the software level. This section focuses on the architectural or system level, as it offers the highest potential in increasing the robustness of digital circuits [Bau08]. Additional details on process, circuit and software techniques are found in [Nic10].

At the architectural level, the introduction of redundant sequential elements is used as a basic principle. Most schemes rely on local redundancy achieved by adding two memory elements, although the implementation at bit granularity limits the achievable efficiency in terms of hardware overhead. One of these memory elements may be a C-element or an additional voter is required (Figure 3.2).

The composition of a reliable system out of unreliable components can be achieved by *Triple Modular Redundancy (TMR)* [vNeu56; LV62]. It introduces space redundancy by triplicating the basic building blocks and combines their results using a synchronous majority voter. It has been shown, that the application of TMR to latches as depicted in Figure 3.2-a in combination with delayed clocks is effective in mitigating transient glitches at the input, and especially SEUs affecting the latches [ME02; CO07]. The hardware overhead of such a scheme is around +400% (two latches plus voter) and is further increased by the need for multiple clock trees if time redundancy is employed. If the voter is triplicated in order to improve the detection probability as in [WWW+03], additional overhead is introduced. Moreover, the delay of the data path is increased due to the voter.

The *Built-In Soft-Error Resilience (BISER)* scheme combines one latch and the voter of TMR into an asynchronous majority voter, the *C-Element* (Figure 3.2-b) [MSZ+05; ZMM+06]. The C-Element [MB57; SEE98] retains its previous output value until both inputs have a common logic value. Thus, any transition on a single input is filtered and all single bit upsets affecting the latches are mitigated. In order to reduce the hardware overhead, the authors propose to reuse the scan portion to implement one of

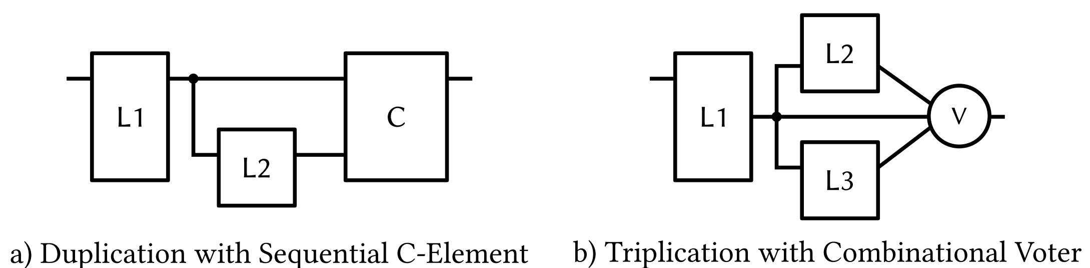

Figure 3.2.: Principle of Robust Latch Design.

the two remaining latches. This might not always be possible, especially when partial scan is used where not all registers are equipped with scan design. Even in presence of scan, if an L1/L2* scheme is employed, the scan portion is already allocated during system mode. Reusing a scan latch during functional operation implies that the scan latch can be operated at the same speed as the main latch in order to not sacrifice performance. If the scan portion is not designed for at-speed scan due to power and area constraints, additional effort might be involved [Muk08].

A variety of other hardened latches and flip-flops exist that increase the resilience to soft errors by means of filtering or local redundant design. The *dual-interlocked storage cell (DICE)* [CNV96] employs two redundant latch sections. The sections are cross-coupled in a way that the data in the uncorrupted section provides specific state restoring feedback to recover the corrupted data. The *delay filtered DICE (Df-DICE)* [ND05] combines DICE with a SET-filtering technique for every input. While this adds immunity to SETs with a duration smaller than the introduced delay, the data path is also delayed and performance is reduced. The *SER-tolerant path-exclusive latch* [HKW+04] adds a redundant standard clocked keeper to a path-exclusive latch and divides all other internal gates and the output buffer to balance the drive strength and load on the redundant storage nodes. Although the data path speed is retained, spurious glitches at output Q during recovery cannot completely be avoided. The *Schmitt trigger (ST) latch* [LKL11] employs the large voltage hysteresis of a Schmitt trigger to mask transients in the latch feedback loop. While it offers a low power consumption and area overhead, the added data path delay limits the performance. The *feedback redundant SEU-tolerant (FERST) latch* [FPME07; FMEP09] uses a redundant feedback path and three C-elements to filter transient errors at most internal nodes. However, the output nodes are still vulnerable while the two C-elements between the input D and the output Q introduce additional delay and thus reduce performance. By delaying one of the two feedback paths, the immunity to SETs can be increased [FMEP09]. The *high-performance SEU tolerant (HPST) latch* [Hua14; HLH15] builds upon the FERST latch and optimizes the power-delay product. To this end, the C-element at the output is clock gated while new data is latched and the output Q is directly driven by the input D. Although this improves the power-delay product, the area overhead is comparable to the high overhead of a TMR latch.

In the following, two solutions will be discussed in more detail. These include the RAZOR architecture ([EKD+03], Figure 3.3) and the GRAAL architecture ([Nic07], Figure 3.4).

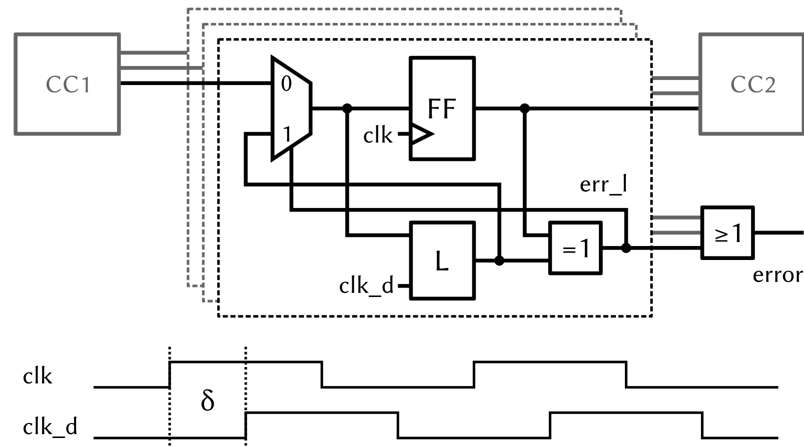

Figure 3.3.: RAZOR Architecture (adopted from [EKD+03]).

The *RAZOR* architecture has been designed to detect propagation delays due to voltage scaling but can also detect and correct SETs under some conditions [EKD+03]. It targets an edge-triggered design style and combines space and time redundancy as depicted in Figure 3.3. Each bit of the sequential circuit part (dotted box) is extended with a redundant latch that samples the logic value from the combinational circuit part *CC1* at a second timepoint controlled by the delayed clock *clk_d*. Thereby, SETs are guaranteed to be detected as long as two conditions hold: The duration of the transient is smaller than the phase shift δ between the two clocks *clk* and *clk_d*; and all paths in the combinational circuit parts have a delay of at least δ. A local error signal *err_l* is derived by comparing the outputs of the flip-flop and the latch by an XOR gate. The error signal controls the multiplexer in order to copy the latch value to the flip-flop. Note, that the multiplexer is usually merged with the feedback loop contained in the flip-flop to minimize additional delay on the data path. In addition, an OR-tree is used to derive a global error signal over all bits to indicate the presence of errors, restore complete registers instead of a single bit, and gate the clock of all other registers in the module to hinder the error effect from propagation.

In case of any difference between the flip-flop and latch values, the local error signal is 1. Thus, the flip-flop value is restored from the redundant latch. If the discrepancy is caused by a SET (with a duration $< \delta$), the latch contains the correct value which was sampled at a sufficiently late timepoint. If the transient lasts longer than δ, both, the flip-flop and the latch, contain the wrong value and the SET is not detected. For SEUs directly affecting the sequential elements two cases need to be distinguished: If

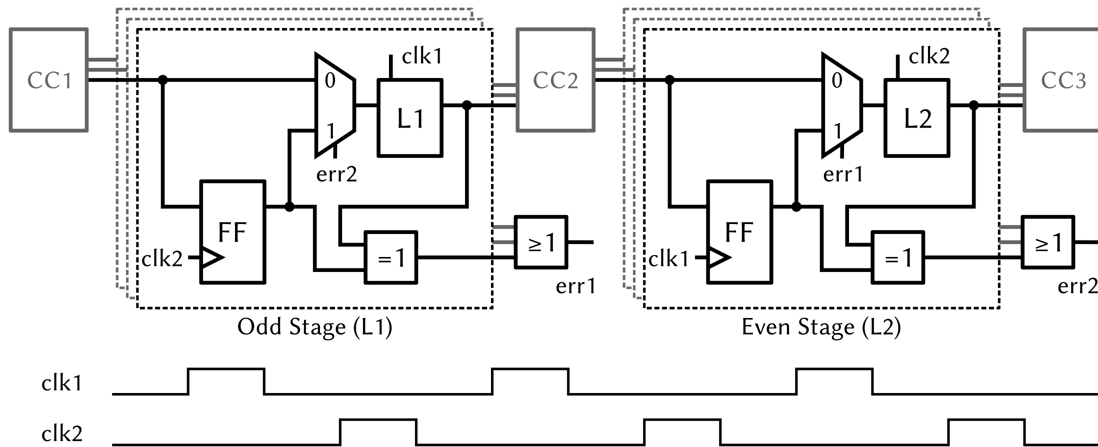

Figure 3.4.: GRAAL Architecture (adopted from [Nic07]).

the flip-flop is hit, the correct value is restored from the unaffected latch and the upset is corrected. If the redundant latch is struck by the upset, the employed redundancy correctly detects a discrepancy but is not able to localize its origin in the flip-flop or the latch. Hence, a false correction is performed.

The *Global Reliability Architecture Approach for Logic (GRAAL)* eliminates the stringent timing requirements inherent to the RAZOR approach and allows to correct SEUs [Nic07]. It also combines space and time redundancy, but targets a level-sensitive design style as shown in Figure 3.4. Latches are used on the data path, are grouped into odd (*L1*) and even (*L2*) stages, and are controlled by two non-overlapping clocks *clk1* and *clk2*. In each stage, the functional latches are equipped with a redundant flip-flop, a comparator to derive a local error signal and a multiplexer to restore the flip-flop value to the latch. The correction works as follows: Assume without loss of generality a SEU in the L2 stage that flipped the value of the flip-flop. Instead of restoring the value of the affected even stage, the error signal *err2* is used to restore the latches in all odd stages. Thereby, any error effect is eliminated that already propagated to the consecutive odd stage. In the second half of the clock cycle, the values of all even stages (including the affected stage) are recomputed from the (correct) state of the odd stages. Consequently, the upset is corrected in one additional clock cycle. The correction is also effective for SEUs affecting all other sequential elements as well as SETs. In [YNA09] a detection scheme based on the GRAAL architecture is presented, that combines latch-based design and time redundancy. While being effective for the detection of timing errors, transients, and upsets, the correction needs to be performed at a higher abstraction level, e.g. by instruction-level retry.

The *RAZOR-II* scheme focuses the hardware support on the in situ detection of errors (both, SETs and SEUs) in registers, where spurious transitions in the state-holding latch node are detected as errors. The correction is performed by architectural replay, which requires several clock cycles to recompute the correct result [BKL+08; DTP+09]. A concrete application and implementation for a 32-bit ARM processor was shown in [BDS+11]. The *Bubble RAZOR* scheme addresses the significant hold time constraints introduced by the RAZOR technique. Error detection is based on duplication with comparison in two-phase latch designs, which is combined with a local replay mechanism for the correction [FFK+12; FFK+13].

The recently presented *RAZOR-Lite* scheme also just supports error detection with dedicated hardware. In contrast to earlier versions, the detection is implemented by exploiting internal signals of the used flip-flops, thereby reducing the influence on the data path, such as additional load and delay. Moreover, the presented scheme results in further reduced area and energy requirements [KKF+13].

The time redundant parity proposed in [PNL11] uses information redundancy implemented by a parity tree to detect SEUs. Thereby a localization of the affected bit is not possible and the correction is performed by recomputation.

The protection with Hamming codes was proposed in [HCB95]. The sequential elements are partitioned into registers, which are individually equipped with an encoder and decoder. Thereby, single bit upsets in the register bits are detected, localized, and corrected at the register output. Soft errors can accumulate if write operations are infrequent, as the data word stored in the register is not corrected. The results indicate, that the use of registers with 6 or 7 bits results in a lower area overhead compared to TMR. In [HML+02] a register file with register sizes of 8 or 16 bit is protected with a Hamming code, where the en- and decoder is shared among all registers. While being able to detect SEUs and correct the register output, the scheme incorporates a high area overhead. The use of information redundancy for a whole register is promising, but needs to be carefully designed and implemented to limit the area overhead associated with the code computation, storage, and checking to an economic degree.

Most schemes for sequential elements embedded in random logic introduce space redundancy for each bit of a register. This enables the detection of single event upsets by comparing the original and the redundant bit value. The correction is performed by voting, restoring the correct value from the redundant copy, or architectural replay.

The properties of the discussed solutions can be summarized as follows:

▷ Majority voting has negative impact on the timing behavior of the data path, even when no error occurs.
▷ Space redundancy combined with majority voting results in a high hardware overhead.
▷ Restoring the correct value from a shadow element implies a better, but not negligible hardware overhead.
▷ If the correct value is generated by recomputation, the hardware overhead associated with the correction is avoided, but the runtime is extended considerably if upsets occur.
▷ The use of information redundancy proposed so far allows the area efficient detection of SEUs if a parity bit is used. As a localization is not possible, the correct value needs to be recomputed.

3.2. Test Access

The growing circuit complexity revealed some of the problems inherent to the serial scan paradigm such as high test application times, test data volume and test power consumption. Two solutions that are able to deal with those problems will be detailed in the following: *Test data compression and compaction*, which builds upon the classical scan paradigm and *Random Access Scan*, an alternative architecture providing individual access to the sequential elements.

3.2.1. Test Data Compression and Compaction

Test *compression* and *compaction* takes advantage of the small number of specified bits (care bits) in a test set. Therefore, the design is modified with additional on-chip hardware attached to the scan chains (Figure 3.5). At the input-side the test stimuli from the tester are decompressed and loaded into the chains. At the output-side the test responses are compacted before being delivered to the tester. The advantages associated with such an *embedded deterministic test (EDT)* architecture are twofold: Test volume is reduced as the test data is stored in compressed form on the tester. Test time is reduced as more internal chains can be driven with a given amount of tester channels, thereby enabling the use of shorter chains requiring less shift cycles.

Many different lossless compression techniques exist for the input vectors, which rely on one of the following three basic principles [Tou06]:

- ▹ Applying the same value to multiple scan chains in *broadcast-scan-based schemes*,
- ▹ data compression codes used in *code-based schemes*, or
- ▹ linear operations implemented as exclusive-OR networks or LFSRs in *linear-decompression-based schemes*.

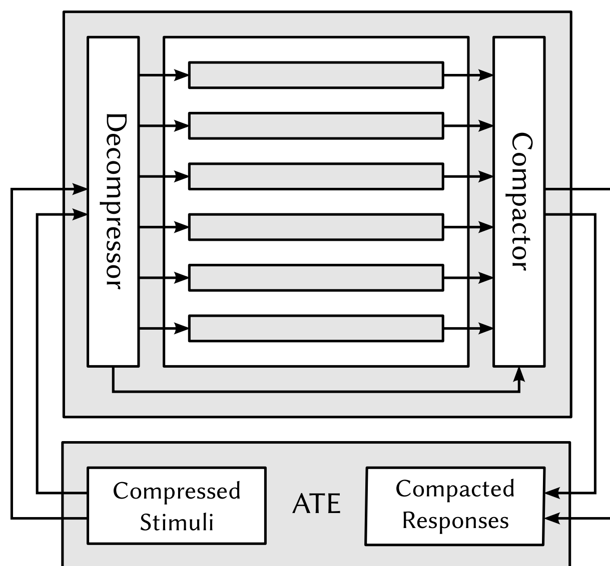

Figure 3.5.: General Embedded Deterministic Test (EDT) Architecture (adopted from [RTK+02]).

The lossy compaction of test responses at the output side is either performed in the time or space domain:

- ▹ *Time compaction* or infinite-memory compaction employs linear finite state machines, such as linear feedback shift registers (LFSR) or multiple input shift registers (MISR). The test responses are compacted into a small signature and high compaction ratios are reached. Unknown values [1] in the test responses corrupt the signature and render the test useless.

[1] An unknown state of signals during the design process, e.g. due to incompletely specified models, which is indeterminable during test generation/simulation but a valid binary value during test.

- ▷ *Space compaction* uses combinational circuits built of exclusive-OR gates to reduce the amount of required outputs. The reachable compaction ratios are lower as compared to time compaction, whereas unknown values in the test responses can be handled.
- ▷ *Convolutional compaction* or finite-memory compaction combines space compaction with a finite memory without feedback.

The embedded deterministic test architecture in [RTK+02; RTKM04] uses a sequential linear decompressor, that is dynamically reseeded by injecting a fixed amount of free variables shifted in from the tester during decompression. In combination with a linear spatial compactor, test data volume is reduced by up to 25 X. If the convolutional compaction from [RTWR03; RTWR05] is used at the output side, compression ratios in excess of 100 X can be reached as long as the decompressor is considered during test generation and the test responses contain a low amount of unknown values. The X-compact scheme in [MK02; MK04] focuses on the response compaction by combinational circuits in presence of unknown (X) values. In combination with a linear decompressor at the input side, arbitrary deterministic patterns can be applied [MK03; MK06].

3.2.2. Random Access Scan

An alternative architecture to increase testability is called *random-access scan (RAS)* [And80]. It allows to address individual flip-flops during test mode similar to the access provided in random access memory (RAM) blocks. Therefore, a general RAS architecture organizes the flip-flops in a two-dimensional array (Figure 3.6).

During functional mode (*TC=0*) all flip-flops latch data from the combinational logic under the control of the clock *CLK* and the flip-flop outputs feed into the combinational logic. A typical storage element used in the addressable array, called a *RAS cell*, is given in Figure 3.7.

During test mode (*TC=1*) individual flip-flops can be read and written. To select a flip-flop, its x- and y-addresses are serially loaded into the two *address shift registers (ASR)* with an additional clock *ACLK*. For a total amount of $n \times m$ bits organized into n columns and m rows, the address sizes are $\lceil log_2 n \rceil$ respectively $\lceil log_2 m \rceil$. The address decoders enable the x- and y-select signals for the addressed flip-flop, while all other select signals stay 0. As only single flip-flops can be addressed, the scanout

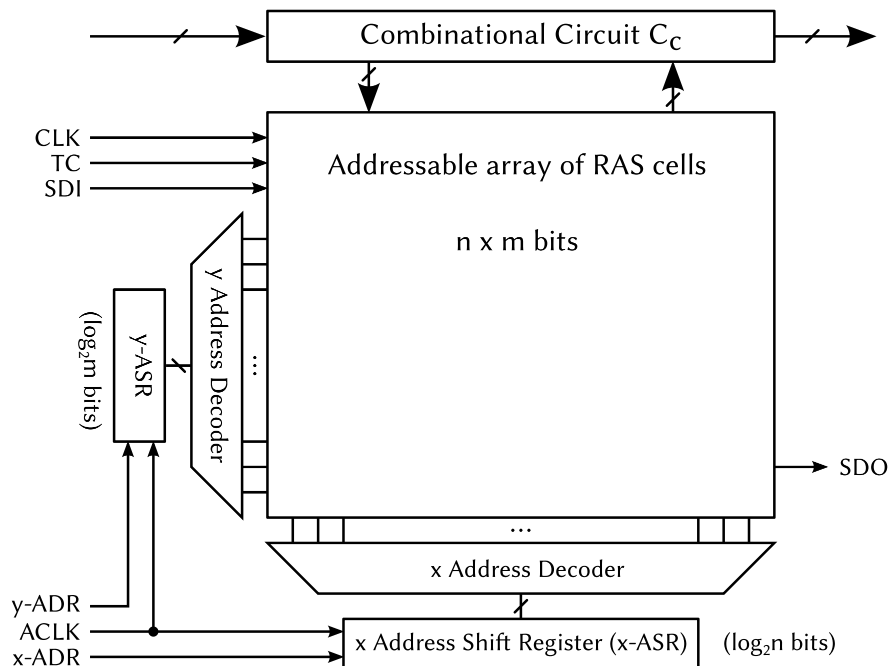

Figure 3.6.: General Random Access Scan Architecture.

signals *SO* of all RAS cells are tied together to the global scanout pin *SDO*. As soon as a RAS cell is selected, its value is observable at the SDO output (independent of the value of the mode signal *TC*). To write a cell during test mode (*TC=1*), the x- and y-select signals of the addressed flip-flop are 1, and the value of *SI* is latched in the flip-flop by *CLK*. All scanin inputs *SI* of flip-flops are connected to the global scanin pin *SDI*. As the select signals of not addressed flip-flops are not conjointly 1, the clock signal *CLK* is blocked and *SDI* is not latched in other flip-flops.

RAS is able to reduce the test time associated with controlling and observing the flip-flop states, but incorporates a large overhead [BA00]. The address decoders and shift registers require additional gates and thereby area. In addition to the clock and select signals, three signals need to be connected to every RAS cell: *TC*, *SDI*, *SDO*. Thereby, significant routing effort and interconnect resources might be required as the RAS cells are distributed within the circuit layout to minimize the impact on circuit delay during functional mode. Moreover, each RAS cell itself incorporates additional gates compared to the scan cells used in scan design. While the RAS architecture has been used in practice [Ito90], it did not gain popularity, perhaps due to the large overhead involved [BA00]. A series of publications built upon the basic RAS idea in order to reduce the overhead and improve the test time, volume, and power.

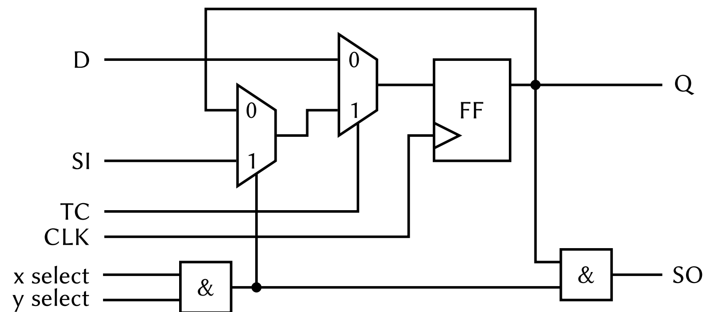

Figure 3.7.: Multiplexer-based Addressable Random Access Scan Cell.

In *Progressive Random Access Scan (PRAS)* [BS05a], the RAS cell consists of a transmission-gate based flip-flop, where the master latch is extended by two additional transistors. Thereby, a SRAM cell is formed which is accessed over sense lines as in a regular SRAM. Read operations are performed in parallel for complete rows and compacted into a signature with the help of a MISR implemented together with the sense amplifiers. While reducing the area of the RAS cell, additional overhead arises from the doubled amount of column select lines (positive and negative polarity), the column drivers and the sense amplifiers.

The idea behind *Toggle RAS* is to invert the value stored in a RAS cell instead of writing it [MAS05a; MAS05b]. The inversion is implemented by a toggle-flip-flop that is activated whenever a cell is addressed in order to be read. Thereby, the global SDI network driving the SI inputs of all cells is eliminated. The combination of toggle RAS and progressive RAS depicted in [AAS+10] further reduces the hardware overhead by modifying the T-FF based RAS cells. For reading, a combination of sense amplifiers with a MISR is utilized similar to progressive RAS, thereby allowing to read and compact complete rows.

In principle any test set can be applied using Random Access Scan, although it might result in elevated test time and test volume. In RAS, the test time depends on the amount of write and read operations required during test application, whereas the time to conduct each operation is dominated by the required shifts of the address registers. The number of write operations between two test vectors is equal to the Hamming distance between the sequential states of the vectors.

In [BSK04; BS05a] existing test sets are reordered to minimize the amount of address scan operations needed during test application in the RAS architecture. The problem is modeled as a asymmetric traveling salesman problem, where the Hamming distance

between test vectors serves as a cost function. The heuristically determined solution is further improved by identifying don't care values in the test vectors and appropriate filling (before and after the reordering). If multiple bits differ between vectors, the write operations are reordered to minimize address shifts by reusing portions of the scanned addresses. [BSK04] compares the reordered test sets for benchmark circuits equipped with a single chain for serial scan and a single address register for RAS. The reordered test sets show a test time speedup between 1.06 X and 5.3 X, a test volume reduction by between 25.27 % and 82.26 % and a test power (defined by the number of gate outputs switching) of below 1 % compared to serial scan. In [BS05a] the reordered test sets are evaluated w.r.t. multiple (3 to 4) scan chains. A speedup between 1.9 X and 4.6 X and a volume reduction between 10.4 % and 62.6 % are reported.

In [BS05b], a test generation procedure for Progressive RAS is described. It iteratively generates test patterns, where each test pattern targets the faults with the highest probability of being detected by the current state of the RAS cells. Thereby, every generated test pattern has a low Hamming distance to the current state. In comparison to the test set reordering in [BS05a], the test time and volume are reduced by additional 20 % on average, with a maximum reduction by up to 50 % for two circuits.

3.3. Combined Solutions

Two necessary tasks related to the reliability of digital circuits have been identified. Test to assess the reliability of every produced chip at the beginning of the lifetime and fault tolerance to cope with soft errors during the operation. Thereby, on-chip infrastructure is used to increase testability by providing test access and to efficiently detect and correct soft errors. A variety of solutions exist in both areas that can be combined and implemented orthogonally. Using a common infrastructure that is able to provide test access during test mode of a circuit and fault tolerance during system mode offers the potential of a reduced hardware and area overhead.

The BISER scheme [MSZ+05; ZMM+06] discussed in Section 3.1.2 reuses portions of the test infrastructure to implement fault tolerance. The AVERA cell in [ZMT+07] extends the scanout portion of the BISER scheme with an additional compaction of the test responses. During test both, BISER and AVERA, allow to observe the sequential elements, while controlling them is not possible. During operation, single event upsets affecting the latches are filtered by the C-element, but not corrected.

In [GBMR06] the area overhead of BISER is optimized by reducing the number of latches from four to three, whereas the system latch is fully controllable and the third latch can be used as a hold latch. Thus, *enhanced scan* is implemented efficiently, which allows to apply arbitrary two-pattern tests by storing a first input vector, shifting in a second vector and applying both vectors in consecutive cycles. The area overhead of the enhanced scan flip-flop with soft error correction from [GBMR06] is further reduced in [NII10]. In [OJC07], instead of a C-Element triple modular redundancy with a majority voter is used for error masking. Thereby, the complete clock period can be protected at the cost of additional area overhead while still providing enhanced scan capabilities. The combination of scan design and local soft-error detection can also be used in the context of irradiation testing as discussed in [YKI+08].

An enhancement of random access scan for soft error tolerance was proposed in [WA10]. The toggle RAS-cell from [MAS05a] contains a feedback path from the slave latch to the master latch that is activated whenever the cell is not addressed. For this reason, the cell is inherently radiation hardened and exhibits reduced soft error rates. Addressed cells are protected either similar to the BISER scheme by employing one additional redundant cell in combination with a C-element or by triple modular redundancy with two redundant cells and a majority voter. In [EMA11; EMAF13] the built-in design for testability resources are reused to recover faulty modules in a coarse-grained triple modular redundancy scheme.

Part II

Fault Tolerance Infrastructure

Chapter 4

Non-Concurrent Detection and Localization of Single Event Upsets

Digital circuits manufactured in recent technologies have to cope with two major challenges aggravated by continued scaling: Strict power and energy budgets and an elevated susceptibility to soft errors. Power consumption is often addressed by clock gating which partitions a circuit into modules and utilizes gated clock trees to disable the clock of individual modules. Thus, switching activity is avoided whenever a module is idle while the sequential state of a module, typically organized in registers, is retained. Consequently, in presence of soft errors with the potential to alter this state, the correctness of data needs to be assured upon leaving such a phase. Clock gated circuits are specifically error prone due to the probability of state corruption being proportional to the length of the gated phase during which errors can accumulate.

The most essential contribution to protect a clock gated module is the detection of spurious modifications of register values. A correction can then be performed by restarting the module's computation with an often acceptable performance impact, especially if errors occur seldom. However, if the error is in addition localized at register granularity, a correction might not always be obligatory, when, for example, the data of an affected register is known as not being used in further computations or if the data was already read before the error occurred.

This chapter is based on the fault tolerance scheme presented in [IWZ08a; IWZ08b] which targets Single Event Upsets during clock gated phases. To this end, individual registers are geared with error detection and the localization of failing registers within a module is enabled. The differentiation between the detection and localization of soft errors and their implementation at different abstraction levels is shown to facilitate a low area overhead. The impact on power consumption is strictly limited by equipping the architecture itself with clock gating.

4.1. Non-Concurrent Architecture

The presented architecture distinguishes between the detection and the localization of Single Event Upsets as shown in Figure 4.1. This two-tiered approach allows to tailor the architecture properties to comply with the requirements posed by the targeted application: The detection and localization of Single Event Upsets during clock gated phases, a low area overhead and a low power consumption during operation.

The sequential portion of a clock gated circuit targeting a level-sensitive design style is depicted in Figure 4.1. The unprotected latches are clustered into N registers $R_1, \ldots, R_N$, where every register can contain an arbitrary number of bits. Without loss of generality it is assumed that every register consists of n bits.

The SEU detection is implemented at gate level as shown in Figures 4.1-b and 4.1-c. Each register R_i is protected by a single parity bit p_i. The parity bit can be computed by a XOR-tree composed out of standard cells (Fig. 4.1-b). By gating the parity tree with the clock disable signal, the parity is only computed during the clock gated phase, thereby limiting the power consumption during operation. Such an implementation incorporates an elevated area overhead as n two-input NAND gates and *(n-1)* two-input exclusive OR (XOR) gates are needed to compute p for a n-bit register. [1]

In order to reduce the area overhead arising from the parity computation per register, the latches and the first level of the XOR-tree are combined and implemented as a new standard cell: The Parity-Pair Latch (PPL). The register is then composed out of *(n/2)* Parity-Pair Latches, *(n/2)* two-input NAND gates and *(n/2)-1* two-input XOR gates (Fig. 4.1-c).

The SEU localization is performed at module level as shown in Figure 4.1-d, where a module-wide checksum is used to identify the affected register. The used modulo-2 address characteristic depends on the register parities $p_1, \ldots, p_N$ and the associated register addresses $1, \ldots, N$ and is implemented area efficiently using XOR gates. Upon entry of a clock gated phase the *reference characteristic C* is computed and stored in the additional register C composed of $\lceil log_2(N+1) \rceil$ latches. During the clock gated phase, the *recomputed characteristic C′* is computed concurrently from the register parities and compared to the reference. The resulting *syndrome S* indicates the presence of changed register parities and provides the address of the failing register.

[1] The gating is not explicitly depicted to highlight the contribution of the Parity-Pair Latch.

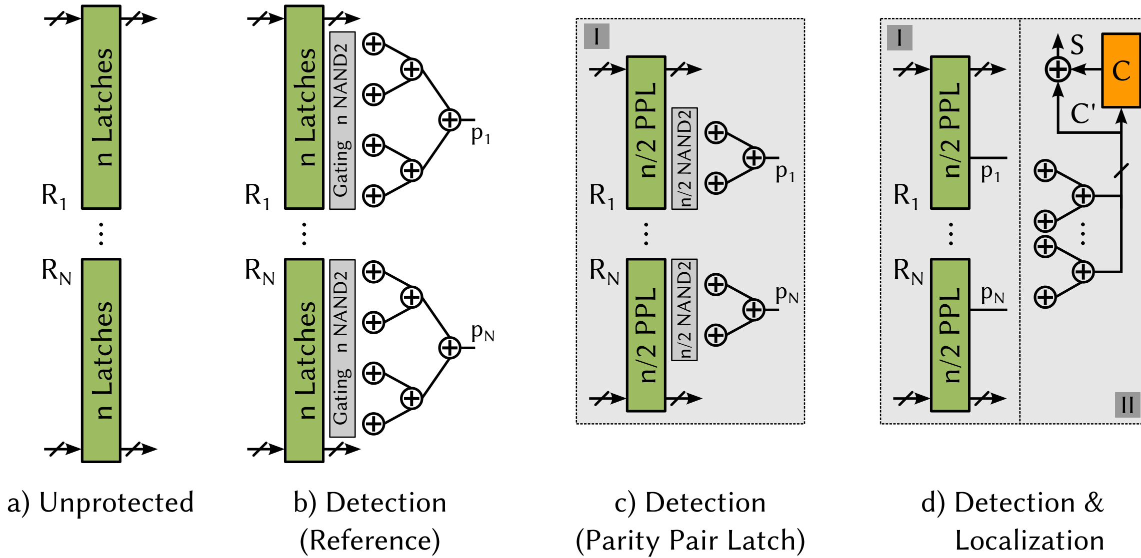

Figure 4.1.: Presented Non-Concurrent Configurations.

The remainder of this chapter is organized as follows. Section 4.2 details the Parity-Pair Latch standard cell and shows how registers can be composed that allow an efficient SEU detection at gate level (Block I in Figures 4.1-c and 4.1-d). Section 4.3 depicts how the register parities are encoded by an error detecting code and how the code computation is implemented in an area efficient way to allow the localization of the failing register (Block II in Figure 4.1-d). Finally, both parts are evaluated experimentally in Section 4.4 before concluding the chapter with a short summary.

4.2. Single Event Upset Detection at Gate Level

In order to detect Single Event Upsets affecting the state of a circuit stored in sequential elements, redundancy in form of additional check bits is employed. To allow error detection, two properties need to hold:

- ▷ The checksum needs to depend on all data bits to be protected.
- ▷ A changing value of any one bit entails a change of the checksum.

4.2.1. Register Parity Protection

The shortest possible checksum to detect Single Event Upsets consists of one additional bit being used for an arbitrary amount of data bits. Often, a parity bit is used, that depicts if the number of data bits with value 'one' is even or odd. For an even parity, a 'one' parity bit is added for an odd number of 'ones' in a register, thereby resulting in an even amount of 'ones' in the total set of bits. Correspondingly, a 'zero' parity bit is employed for an even number of 'ones' (see Definition 2.2.6). [2]

The even parity bit can be calculated effectively as the exclusive OR (XOR) sum of the data bits, which evaluates to 'zero' for an even parity and 'one' for an odd parity. For a register $\vec{R}_i$ with n data bits, the parity p_i is calculated by $n - 1$ two-input XOR gates, which are typically organized in a tree of depth $log_2(n)$.

To detect Single Event Upsets affecting the register data during a clock gated phase, the parity bit is computed upon entry of the clock gated phase and stored in an additional register bit. From now on, this parity will be called the *reference parity* p_i of register R_i. While the clock is gated, the parity tree concurrently computes the *recomputed parity* p'_i. By comparison of the reference and the recomputed parity, the *syndrome* S_i is computed with the help of one additional XOR operation.

$$s_i = p_i \oplus p'_i \tag{4.1}$$

Whenever the reference and the recomputed parity differ, the syndrome s_i is one. Two causes leading to such a non-zero syndrome can be distinguished:

- ▷ A flipped data bit manifests in the recomputed parity p'_i and the syndrome correctly indicates a corruption.
- ▷ The stored reference parity bit p_i is directly affected by a Single Event Upset while the data bits are correct.

For a protected register, all $n + 1$ bits can be affected by a Single Event Upset. Though, the probability of a data bit being affected is $n/(n + 1)$, which is significantly larger than the probability for a corrupted parity bit $1/(n + 1)$. Hence, in rare cases, a Single Event Upset is indicated although all data bits are correct (false detection). [3] Besides

[2] An odd parity bit, the inversion of an even parity bit, could be used in the same manner. In the following, the even parity will be used without loss of generality.

[3] The amount of false detections can be further reduced by increasing the minimum Hamming distance, e.g. by duplication of the parity bit ($n+2$ total bits) and consensus checking (see Section 2.2.3.1).

a slightly reduced system performance such a false detection is not an issue as the triggered countermeasures (e.g. a recomputation) also rectify the corrupted parity.

The power consumption of a module during idle phases is reduced by clock gating. As opposed to normal operation, where the state is written once in a while, the vulnerability to Single Event Upsets raises during long periods of data retention. It follows, that focusing the protection on the clock gated phase offers the highest potential to increase robustness. In order to not sacrifice the power consumption during operation, the parity computation itself is gated whenever the clock is enabled. Therefore, NAND2 gates are employed due to their small area footprint, which are attached to the leaves of the parity tree. A reference implementation of a register with four data bits and a gated parity computation is shown in Figure 4.2.

As the area overhead introduced in order to escalate robustness against SEUs directly relates to the production cost, area is possibly the most important metric besides power consumption and delay. The area of the unprotected register depends on its width in terms of bits as well as the size of the used sequential element and will be used as the area baseline in the following (Eq. 4.2). The area of the reference parity computation is larger due to the additional parity tree (XOR2 gates) and its gating (NAND2 gates) (Eq. 4.3).

$$Area_{Unprotected} = n \cdot A_{Latch} \tag{4.2}$$

$$Area_{Parity} = n \cdot A_{Latch} + n \cdot A_{NAND2} + (n-1) \cdot A_{XOR2} \tag{4.3}$$

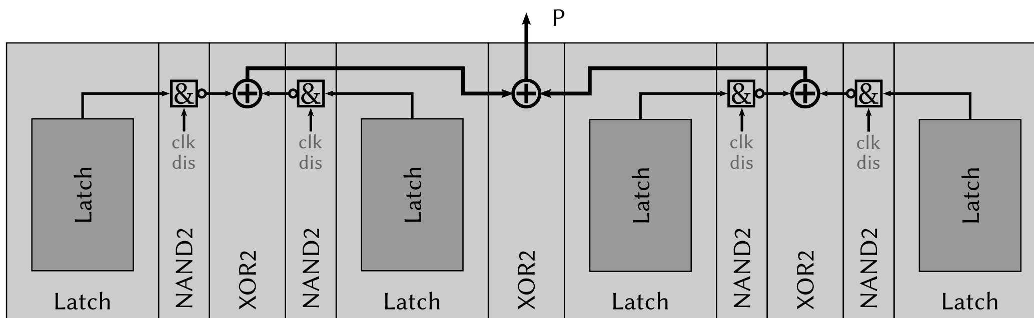

Figure 4.2.: Reference Parity Tree Implementation ($n = 4$).

4.2.2. Area Efficient Register Parity Computation - Parity-Pair Latch

While the parity computation for a register is effectively implemented by a tree of standard cells as depicted in the last section, such an implementation might not be as area efficient as desired. Rewriting Formula (4.3) unveils, that a significant share of the area overhead is spent to implement the first level of the parity computation.

$$Area_{Parity} = \underbrace{n \cdot (A_{Latch} + A_{NAND2}) + \frac{n}{2} \cdot A_{XOR2}}_{\text{first level}} + \underbrace{(\frac{n}{2} - 1) \cdot A_{XOR2}}_{\text{remaining levels}} \tag{4.4}$$

Implementations with much less impact on power, delay and area can be found, if the first level of the parity tree is merged with the latches. Figure 4.3 shows the schematic of the parity computation between two latches from [IWZ08a; IWZ08b], referred to as the *Parity-Pair Latch (PPL)* [4]. Each latch consists of a feedback loop with two inverters (INV1/2 resp. INV3/4) that drives the output Q, as well as two transmission-gates (TG1/2 resp. TG3/4) used to control each latch. To this end, either the feedback loop is disabled and a new value from input D is registered (latch-operation), or the registered value is stored within the enabled feedback loop (hold-operation). The parity computation is especially efficient to implement with two transmission-gates TG5 and TG6, as the latches already provide both polarities of their internal state.

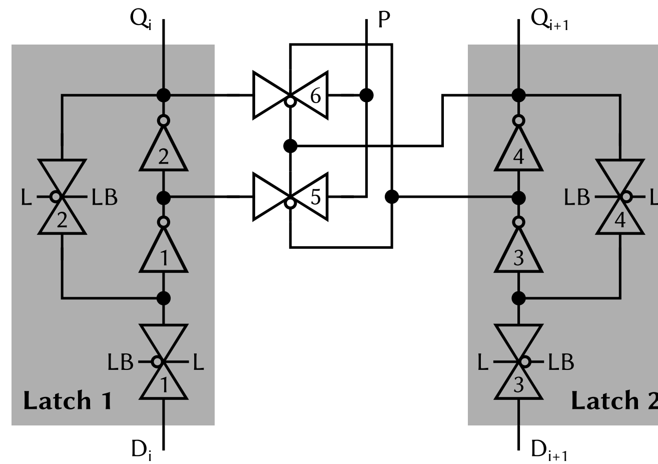

Figure 4.3.: Schematic of the Parity-Pair Latch (PPL).

[4] The Parity-Pair Latch from [IWZ08a; IWZ08b] implements local clock gating internally. The generalized implementation depicted here results in a lower area overhead if clock gating is not desired.

With these Parity-Pair Latches (PPL) a register is formed like in Figure 4.4. In order to reduce the power consumption during operation, clock gating is implemented by accompanying the Parity-Pair Latch cells with standard CMOS NAND2 cells controlled by the inverted clock gating signal. Hence, the parity is only computed during clock gated phases of the module and the switching activity during operation is reduced to a minimum. As the PPL parity computation is performed cell internal, the NAND2 gating cells are connected to the parity outputs of the PPL cells where they also restore the signal levels of the transmission-gate based parity computation.

Compared to a standard CMOS XOR2 with 8 or 10 transistors, the parity computation in the Parity-Pair Latch requires only 4 transistors. The critical path of the PPL is just 4 inverters and 3 pass transistors which is less than three times the delay through a latch, and in the same range as any of the double latch solutions mentioned in Section 3.1.2. As in the reference implementation, the remaining levels of the parity computation are implemented as a XOR tree composed of standard XOR2 cells.

In excess of the advantages with respect to power consumption and delay, such an optimized register parity computation comprises a reduced area overhead (Eq. 4.5)

- ▹ as soon as the area of a Parity-Pair Latch standard cell implementation is smaller than the sum of the replaced equivalent latch and exclusive OR standard cells;
- ▹ in presence of clock gating, as the amount of gate cells at the leaves of the remaining parity tree is bisected.

$$Area_{PPL} = \underbrace{\frac{n}{2} \cdot (A_{PPL} + A_{NAND2})}_{\text{first level}} + \underbrace{(\frac{n}{2} - 1) \cdot A_{XOR2}}_{\text{remaining levels}} \tag{4.5}$$

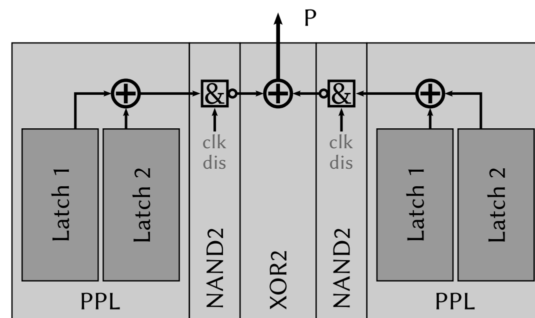

Figure 4.4.: Parity Tree Implementation utilizing Parity-Pair Latches (n=4).

4.3. Single Event Upset Localization at Module Level

The error information of the basic registers has to be passed to module's top level, stored in a register and will be used for error handling. If only error detection is required in order to restart computations, a simple XOR tree may connect all the registers. More effort is needed, if the detection of multiple errors is also desired, or if the error location is required in order to allow more fine-grained recomputation decisions to be taken at module level.

This localization information will be provided by a checksum computed over all register parity bits, which allows a detection and localization of Single Bit Upsets (Block II in Figure 4.1-d). Therefore, upon entry of a clock gated phase, a *reference characteristic* C is computed and stored. Throughout the clock gated phase, the *recomputed characteristic* C' is concurrently derived from the register parities and compared to the reference characteristic.

4.3.1. Modulo-2 Address Characteristic

In embedded memories, transparent online test can be used for error localization (see Section 3.1.1.2, [HWI+02; BNLC06]). The transparent test technique for static and dynamic memory arrays from [HWI+02] has been adapted to random logic in [IWZ08a; IWZ08b]. The *modulo-2 address characteristic* of a bit-oriented memory is computed by a bit-wise XOR of the addresses of those memory cells which contain a '1' (Figure 4.5).

Definition 4.3.1 (Modulo-2 Address Characteristic) *Let $\vec{D}$ be an m-bit wide memory with addresses $A = \{m, \ldots, 1\}$ and let $A_1 := a \in A \mid \vec{D}[a] = 1$ denote the set of all addresses containing a logic* 1*. Then, the modulo-2 address characteristic is defined as*

$$\vec{C} = \bigoplus_{a \in A_1} a \quad .$$

The characteristic is now used to localize the module register affected by a Single Bit Upset during clock gated phases. Let p_i denote the parity of register R_i calculated by the Parity-Pair Latches from the previous section. Let $\vec{P} = [p_N, \cdots, p_1]^T$ represent

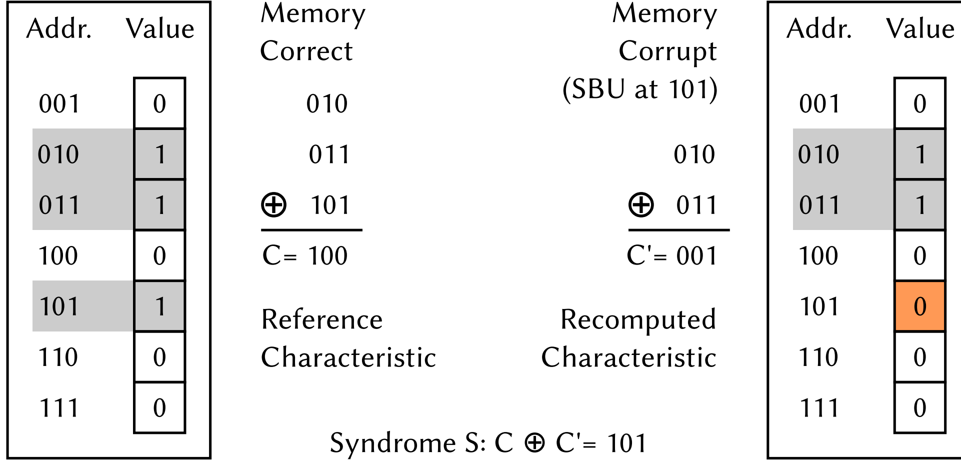

Figure 4.5.: Modulo-2 Address Characteristic.

the parity vector in matrix notation of all N register parity bits in the module, where p_{adr} $(N \geq adr \geq 1)$ references the parity of register R with address adr.

The mapping between parity and characteristic bits corresponds to a Hamming code check matrix (see Eq. 2.5) expressed by the modulo-2 characteristic matrix M.

$$M = \begin{bmatrix} \rotatebox{90}{$adr\ N$} & \cdots & \rotatebox{90}{$adr\ 1$} \end{bmatrix}$$

It consists of l rows and N columns, where N is the number of parity bits and each column contains the binary address adr of the associated parity bit. The size l of the calculated characteristic is defined by the maximum length over all used addresses and depends on N logarithmically.

$$l = \lceil log_2(N + 1) \rceil \tag{4.6}$$

The characteristic $\vec{C}$ is computed by multiplying M with $\vec{P}$.

$$\vec{C} = M \cdot \vec{P} \tag{4.7}$$

To detect an error, the reference characteristic $\vec{C}$ of the original parity bits is computed at the initiation of a clock gated phase and stored in an additional register of size l.

The recomputed characteristic $\vec{C}'$ is then concurrently derived from the parity bits $\vec{P}$ until the end of the clock gated phase. The difference between the reference characteristic $\vec{C}$ and the recomputed characteristic $\vec{C}'$ is called the syndrome of $\vec{P}$.

$$\vec{S} = \vec{C} \oplus \vec{C}' \tag{4.8}$$

If $\vec{S}$ is the all-zero vector no deviation was detected, otherwise $\vec{S}$ contains the address localizing the register bit affected by a *Single Bit Upset (SBU)*.

Example Assume a module with 7 registers and let $\vec{P}$ denote the associated parity bits with values $\begin{bmatrix}0010110\end{bmatrix}^T$. Together with the modulo-2 checksum matrix M, the reference characteristic $\vec{C}$ of size $l = \lceil log_2(7+1) \rceil = 3$ is computed and stored.

$$\vec{C} = M \cdot \vec{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot \vec{P} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Now, a SBU affects register R_5 which results in the inversion of the associated register parity p_5 and leads to the faulty parity vector $\vec{P}' = \begin{bmatrix}0000110\end{bmatrix}^T$. The characteristic is recomputed as $\vec{C}'$ and the syndrome $\vec{S}$ is calculated.

$$\vec{C}' = M \cdot \vec{P}' = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \vec{S} = \vec{C} \oplus \vec{C}' = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

As $\vec{S}$ is not the zero vector, the SBU is detected. Moreover, as $\vec{S}$ contains the address 5, the register affected by the SBU is correctly localized.

The characteristic has some substantial benefits compared with signature analysis and error correcting codes (see Sections 2.2.3.1 and 3.1.1.1) [DSS95; Ham50]. Single error location is especially easy, as the computed syndrome directly provides the address of the affected register. Double errors are always detected, and the overall aliasing probability is 2^{-l}, where l is the number of address bits [WDGS87; WDGS88; DOFR89]. Address '0' is not used, as it does not contribute to the error detection. The overall characteristic can be computed sequentially bit by bit like in Figure 4.5.

4.3.2. Optimal Combinational Characteristic Computation

However, the sequential computation is only appropriate for memory arrays, but in random logic a combinational logic is required. In a straight-forward way this is implemented by a tree whose leaves are the N binary l-bit words $p_i \wedge i$, and each node represents bit-wise XOR. Figure 4.6 illustrates this for 7 registers (N=7) with parity bits $p_1, \dots, p_7$ for $l = 3$.

The number of vertices in the tree of Figure 4.6 is $2^{l+1} - l$, due to address 0 not being used. As the root node has not to be connected two times there are

$$l \cdot (2^{l+1} - l - 1) \tag{4.9}$$

point-to-point connections. A significant amount of hardware can be saved if only a subset of all bits is passed between the levels (Figure 4.7).

To recapitulate: The characteristic C is the modulo-2 sum of all addresses that contain a logic 1 (see Def. 4.3.1). It follows, that the i'th characteristic bit is the modulo-2 sum of all i'th address bits that contain a logic 1 (Fig. 4.6). However, only addresses that have a logic 1 at position i can actually contribute to characteristic bit i. Consequently, as the addresses are a priori known, each characteristic bit i can be computed by a XOR-tree that sums up all parity (data) bits where the i'th address bit is 1 (Fig. 4.7).

In level $k \in \{0, \dots, l-1\}$ there are 2^k vertices and each vertex $v_{k,i}, i \in \{0, \dots, 2^k - 1\}$ corresponds to 2^{l-k} addresses. $c_{k,i} = (c_{k,i,l-1}, \dots, c_{k,i,0})$ is the output of vertex $v_{k,i}$. The k most significant bits $(a_{l-1}, \dots, a_{l-k+1})$ of the addresses of vertex $v_{k,i}$ are identical and the result bits $(c_{k,i,l-1}, \dots, c_{k,i,l-k+1})$ only depend on the parity of the 2^{l-k} leaf vertex register-parities:

- ▷ $(c_{k,i,l-1}, \dots, c_{k,i,l-k+1}) = (0, \dots, 0)$ if the parity is 0,
- ▷ $(c_{k,i,l-1}, \dots, c_{k,i,l-k+1}) = (a_{l-1}, \dots, a_{l-k+1})$ if the parity is 1.

Thus, it is not necessary to compute this vector of k bits. Instead, the generation of the information can be deferred to the successor in level $k - 1$ and only the parity is computed and forwarded:

$$p_{k,i} = p_{k+1,2\cdot i} \oplus p_{k+1,2\cdot i+1} \ .$$

The $l - k - 1$ bits $(c_{k,i,l-k-2}, \dots, c_{k,i,0})$ are computed from the characteristics of the predecessors of $v_{k,i}$:

$$c_{k,i,j} = c_{k+1,2\cdot i,j} \oplus c_{k+1,2\cdot i+1,j} \qquad j \in \{l - k - 2, \dots, 0\} \ .$$

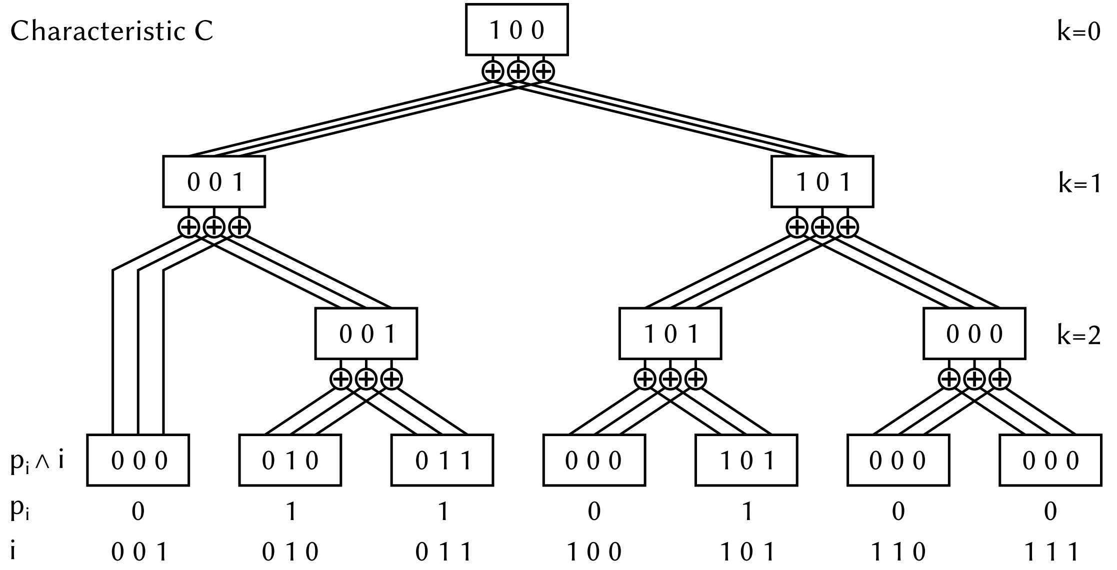

Figure 4.6.: Non-optimal Computation of the Modulo-2 Characteristic.

Finally, $c_{k,i,l-k-1}$ corresponds to the highest address bit that distinguishes the two predecessors of $v_{k,i}$. From the previous observations, it is known that any most significant bit of the characteristic can be derived from the parity bits $p_{k,2\cdot i}$ and $p_{k,2\cdot i+1}$. Here, address bit $l-k-1$ of predecessor $v_{k,2\cdot i}$ is known to be 0, and 1 for $v_{k,2\cdot i+1}$ respectively. It follows:

$$\begin{aligned} c_{k,i,l-k-1} &= (0 \wedge p_{k+1,2\cdot i}) \oplus (1 \wedge p_{k+1,2\cdot i+1}) \\ &= p_{k,2\cdot i+1} \,. \end{aligned} \tag{4.10}$$

Therefore, each vertex on level k has 2 connections to read each parity of the 2 predecessors and $2\cdot(l-k-1)$ connections to read the characteristic of the predecessors. Hence, the number of input connections on level k is $2^k \cdot 2 \cdot (l-k)$.

Since the least significant parity bit in any level does not contribute to the overall characteristic, there is no need to compute this bit (see Figure 4.7). The overall number of point to point connections is:

$$\sum_{k=0}^{l-1} 2^{k+1} \cdot (l-k) - l$$

which is easily transformed into

$$2^{l+2} - 3 \cdot l - 4 \tag{4.11}$$

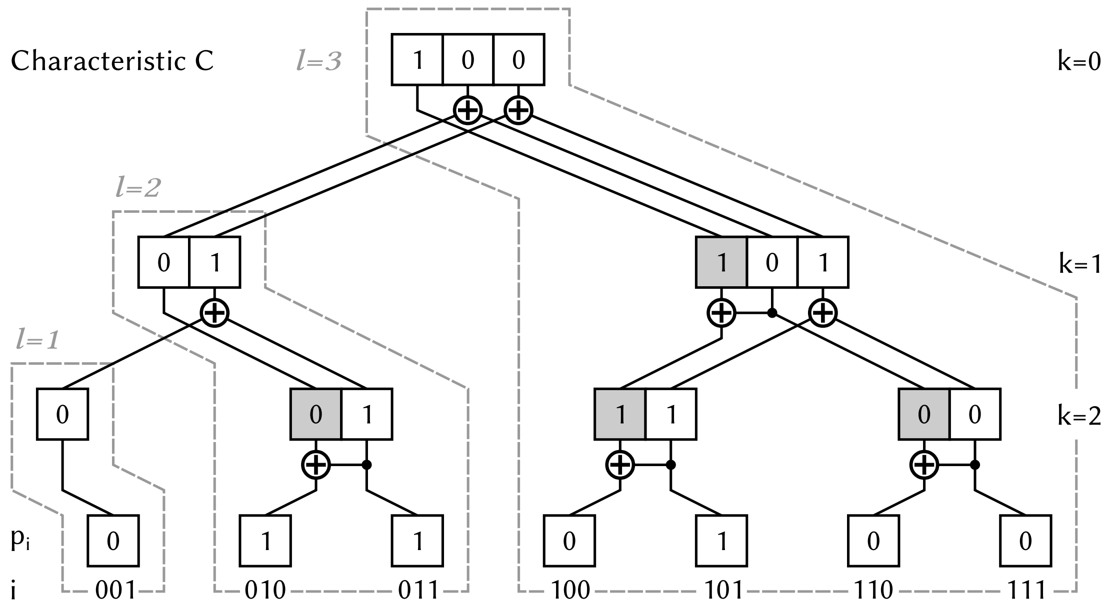

Figure 4.7.: Optimal Characteristic Tree Organization.

and is less than Formula (4.9).

The reduced amount of connections originating from the optimal characteristic tree organization results in a lowered routing congestion in the layout of circuits equipped with the detection and localization scheme. Thereby, alleviating the routing effort and lowering the area associated with routing.

Of even higher importance is the reduction of the cell area required to implement the characteristic tree. The characteristic computation for a single data bit is straightforward as no parity bit is needed due to address 000 not being used. To derive the 2-bit characteristic of 3 data bits with addresses $001, 010, 011$ (the left sub-tree in Figure 4.7 with $N = 3$ and $l = 2$) two additional XOR gates are required in excess of the computation of a single characteristic bit:

- ▷ Characteristic Bit 1 ($k = 1$): To combine the two sub characteristics from the lower level identified by a '1' at the first address position (bits 001 and 011),
- ▷ Characteristic Bit 2 ($k = 2$): To compute the parity of the two bits containing a '1' at the second address position (bits 010 and 011).

In general, the amount of supplemental XOR gates to extend the characteristic from $l - 1$ to l bits is

$$2^l - 2\,.$$

The total amount of XOR gates for a given characteristic size l is then calculated by summation over all levels. The area of the optimal combinational characteristic computation is derived by multiplication with the area of the used two-input XOR standard cell.

$$Area_{Char} = \sum_{2 \le j \le l} (2^j - 2) \cdot A_{XOR2} = (2^{l+1} - 2 \cdot l - 2) \cdot A_{XOR2} \tag{4.12}$$

Example Continuing the previous example, let N be 7. Further, let the characteristic size l be 3 due to Formula 4.6. Consequently, the number of XOR2 gates required to implement the characteristic tree is 8 as depicted in Figure 4.7:

$$\sum_{2 \le j \le 3} (2^j - 2) = 2^{3+1} - 2 \cdot 3 - 2 = 16 - 6 - 2 = 8\,.$$

4.4. Experimental Evaluation

The Non-Concurrent Detection and Localization of Single Event Upsets will be evaluated in the following. The description of the experimental setup is followed by the results for the detection at gate level building upon the Parity-Pair Latch. Finally, the results regarding the modulo-2 address characteristic used to facilitate the localization at module level are discussed. More details on the used tools are found in Section A.1. Tabulated results are provided in Section A.3.1 where appropriate.

4.4.1. Experimental Setup

The Parity-Pair Latch schematic from Section 4.2.2 is implemented at layout level as a new standard cell. The cell is designed according to the design rules and electrical rules of the FreePDK *Process Design Kit* [SCW+07], which also lays the foundation of the Nangate *Open Cell Library (OCL)* [Nan11]. The cell layout is checked for consistency with the schematic during *Layout-vs-Schematic (LVS)*. Afterwards, *Physical Extraction (PEX)* is performed to obtain a transistor netlist modeling all parasitic layout effects at electrical level.

The Parity-Pair Latch transistor netlist, as well as the reference implementation using Open Cell Library gates, are simulated at the analog level using *SPICE* [NP73] in order

to derive the timing behavior and power consumption. In all simulations, the input is driven by a rising and a falling edge with a slew rate of $22\,V/ns$; which implies that the input signal reaches the nominal voltage of $1.1\,V$ respectively $0\,V$ after $50\,ps$. An inverter (OCL INV_X1) is connected to every output as a load. For the output transition delay, the logical '0' is detected below 10 % and the logical '1' is recognized above 90 % of the nominal voltage. In contrast to the 30 % and 70 % used within the Open Cell Library, these percentages are pessimistic and underestimate the reachable delays.

The standard cell is added to a new standard cell library during *Library Characterization*, which determines the electrical properties for a much larger space of operating points than the previous manual analog simulation. As the new library comprehends full compatibility to the Open Cell Library (same cell height and operating voltage) the detection of Single Event Upsets is evaluated at gate level. Therefore, two variants of the parity computation for a single register, a reference implementation and the presented solution employing Parity-Pair Latches, are implemented in *VHDL*, synthesized and compared with respect to their area overhead. Finally, the localization across multiple registers is implemented in VHDL, added to both variants, synthesized and evaluated across different register sizes and quantities.

4.4.2. Single Event Upset Detection at Gate Level

The overhead to implement the Single Event Upset detection at gate level consists of the Parity-Pair Latch (PPL), the remaining XOR gates to compute the register parity and the associated wiring. The properties of a Parity-Pair Latch standard cell implementation will be discussed first, followed by the results regarding the parity computation for a single register.

4.4.2.1. Parity-Pair Latch Standard Cell

The Parity-Pair Latch is implemented as a new standard cell using a full custom design style. In the following, this basic building block of the presented Single Event Upset detection will be evaluated with respect to three metrics. The area of the standard cell in order to quantify the achievable hardware overhead reduction; the timing behavior to substantiate an accelerated parity computation; as well as the power consumption and energy to depict an increased efficiency during operation.

Standard Cell Area

The layout of the Parity-Pair Latch standard cell is shown in Figure 4.8. The order of the gates from the schematic (Fig. 4.3) from left to right is as follows: TG1, TG2, INV1, INV2, TG6, TG5, INV4, INV3, TG4, TG3. The cell height is - as in the Open Cell Library - 1.4 μm. Employing the same cell height ensures interoperability with arbitrary standard cells from the Open Cell Library; and especially enables the use of OCL XOR2 cells to implement the remaining levels of the register parity computation. The Parity-Pair Latch standard cell has a width of 2.66 μm and a total area of 3.724 μm^2.

Implementing the same functionality using OCL standard cells requires two high enable latches (DLH_X1) and one exclusive OR (XOR2_X1). With an area of 2.926 μm^2 for the OCL DLH_X1 latch and 1.596 μm^2 for the OCL XOR2_X1 cell, the resulting area adds up to $2 \cdot 2.926\ \mu m^2 + 1.596\ \mu m^2 = 7.448\ \mu m^2$.

In summary, in contrast to the reference implementation, the newly designed Parity-Pair Latch standard cell PPL_X1 requires just half the area.

Timing Behavior

In order to measure the delay of the Parity-Pair Latch cell (Sec. 4.2.2) and to quantify the improvement with respect to a straight forward implementation solely utilizing Open Cell Library cells, both alternatives are simulated at the circuit level using SPICE.

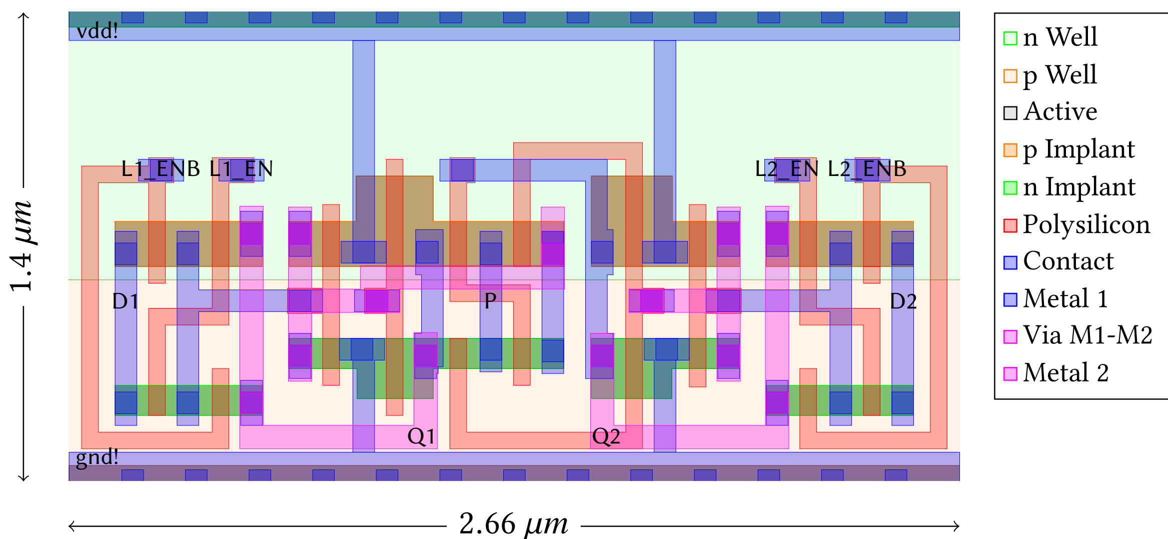

Figure 4.8.: Layout of the Parity-Pair Latch Standard Cell PPL_X1.

The OCL reference implementation of the PPL functionality consists of a transistor netlist composed out of of two DLH_X1 high enable latches and one XOR2_X1 exclusive OR connected to the two latch outputs (left half in Figure 4.2 without NAND2 gates). The Parity-Pair Latch transistor netlist is identical to Figure 4.3. The delays from the D1 input to the Q1 output respectively the parity output P are determined for both netlists. For the PPL netlist the delay from D2 to Q2 and P is considered in addition, as both transmission gates of the XOR function are driven by the second latch while their drains are connected to the first latch. The second latch is transparent throughout the simulation and propagates a logical '0'.

Figure 4.9 shows the simulation results for the OCL implementation. For a falling transition at D1, the Q1 output reaches 0.11 V after 98.76 ps, for a rising transition 0.99 V are reached after 98.63 ps. The parity output P needs 130.99 ps for the falling and 136.45 ps for the rising transition.

Figure 4.10 depicts the results for the implemented PPL standard cell. The D1-to-Q1 delay for the falling transition is 72.4 ps and for the rising transition 76.36 ps. The D1-to-P delays are as low as 74 ps (falling) and 82.42 ps (rising), which is significantly faster as in the OCL reference implementation. As the implementation of the XOR function in the PPL cell is not symmetric, the transition at input D2 is simulated in

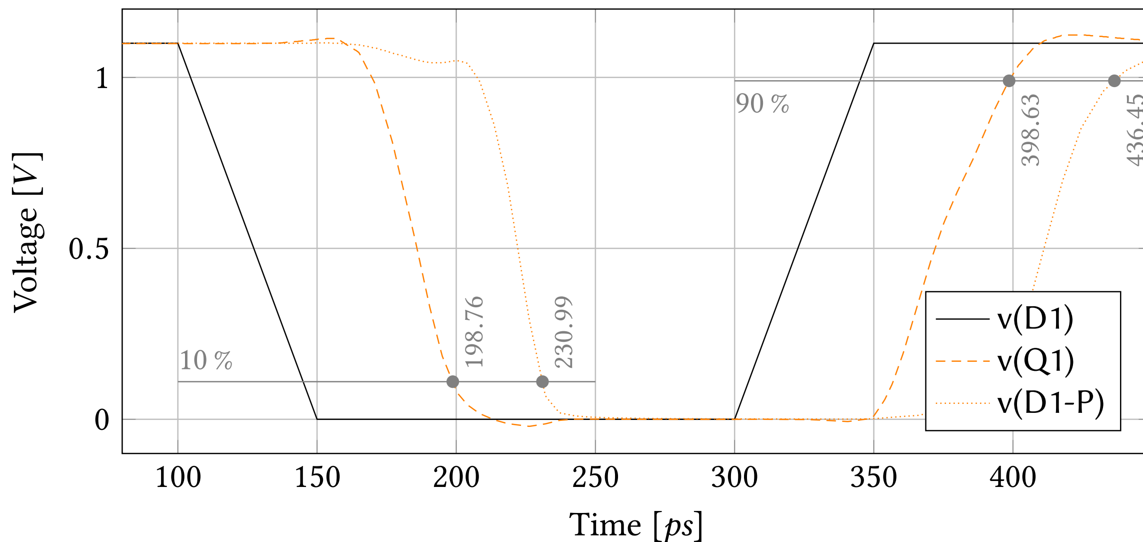

Figure 4.9.: Timing Behavior of the OCL Parity-Pair Latch Reference Implementation (DLH_X1 and XOR2_X1): D1-to-Q1 and D1-to-P Delay.

addition. With D2-to-Q2 delays of 69.33 *ps* (falling) and 69.34 *ps* (rising) as well as D2-to-P delays of 67.74 *ps* (falling) and 73.39 *ps* (rising) the parity even outperforms the latch output and all delays are lower than for the D1 input.

In summary, the Parity-Pair Latch standard cell is faster than the reference implementation for all measured delays, with a considerably accelerated parity computation.

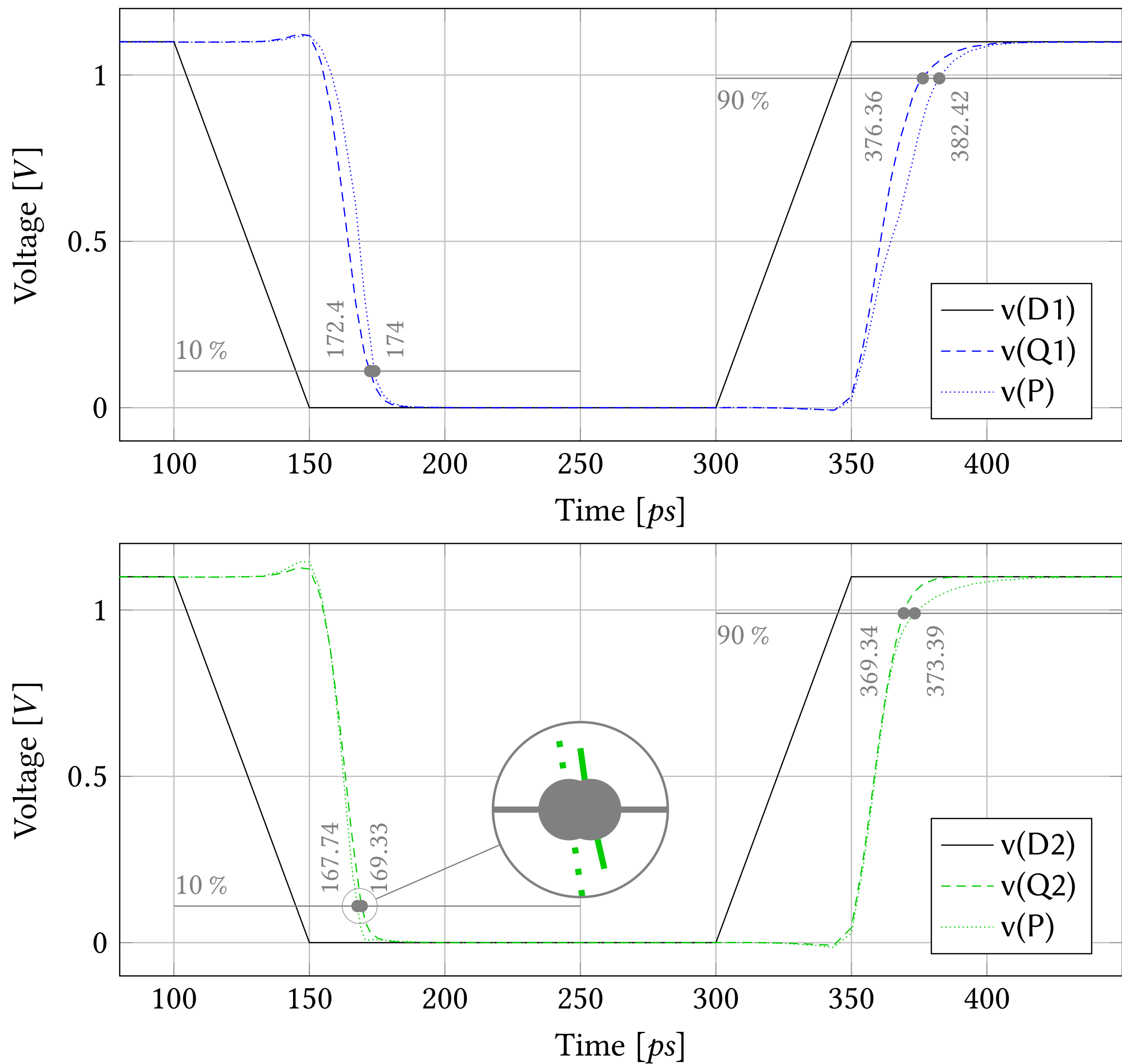

Figure 4.10.: Timing Behavior of the Parity-Pair Latch (PPL_X1): D1-to-Q1, D1-to-P and D2-to-Q2, D2-to-P Delay.

Power Consumption and Energy

During the analog simulation, a transient analysis of the power consumption is performed for both circuits. The power consumption of the reference implementation (depicted in orange in Figure 4.11) shows two major peaks for each transition, which can be attributed to the switching of the contained latch and the exclusive OR. For the Parity-Pair Latch, only a single peak is visible due to the integrated parity computation and power is consumed in much smaller timeframes for both transitions.

The peak power is reduced for both transitions in comparison to the peak power of 112.2 μW for the reference implementation. For the Parity-Pair Latch, the peak power is 91.33 μW for a transition at the D2 input, and 84.29 μW for the D1 input. Consequently, the peak power is reduced by 18.60 % and 24.87 % respectively.

For the average power, the reduction is even higher due to the steeper and shorter power consumption of the Parity-Pair Latch. The average power consumption of the reference implementation during the simulated 500 ns is 23.40 μW. In contrast to this, the Parity-Pair Latch has an average power of 8.14 μW for a transition at the D1 input and 7.86 μW for the D2 input, thereby reducing the average power by 65.21 %, respectively 66.41 %.

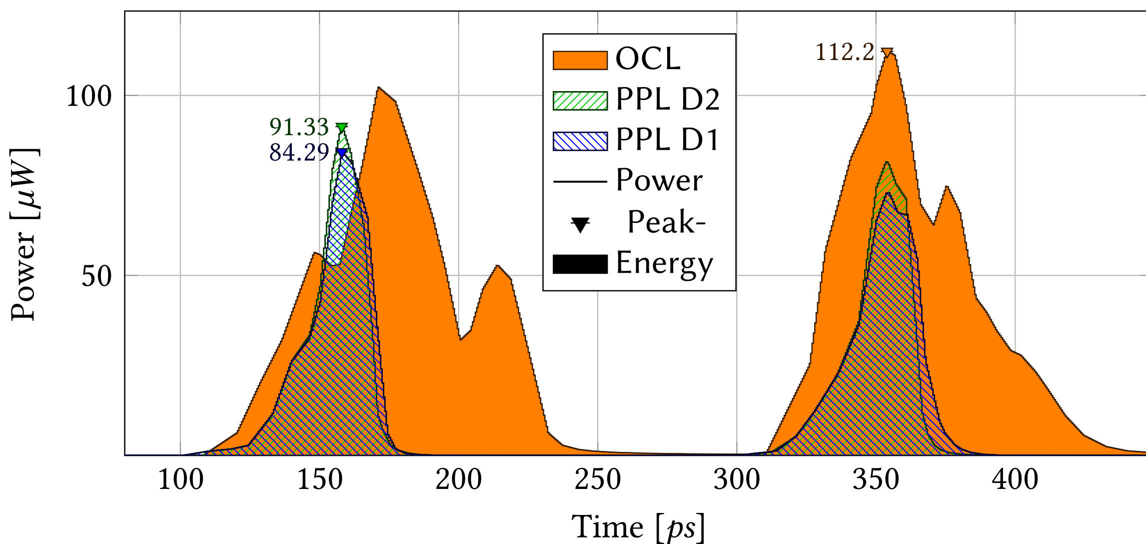

Figure 4.11.: Power and Energy of the OCL Parity-Pair Latch Reference Implementation (DLH_X1 and XOR2_X1) and the Parity-Pair Latch (PPL_X1).

The energy (area under each curve in Fig. 4.11) of the reference implementation is 116.99 pJ (OCL PPL). The energy of the Parity Pair Latch amounts 40.71 pJ (PPL D1) and 39.27 pJ (PPL D2) and is reduced by the same percentage as the average power due to the identical simulation time.

In summary, these results reassure a reduced peak and average power consumption, as well as a significant energy reduction for the Parity-Pair Latch.

4.4.2.2. Parity Computation for a Single Register

Typically, a register contains more than two bits. To this end, several Parity-Pair Latch cells can be combined to form larger registers as depicted in Figure 4.1-c.

Area Overhead

The area required for such an implementation was already discussed in Section 4.2 and will now be evaluated. Figure 4.12 depicts the normalized area after synthesis as a function of register size for three different registers that have been implemented using VHDL. The *unprotected register* consisting of DLH_X1 latches which serves as the area baseline (Eq. 4.2), the *reference parity protection* exclusively using Open

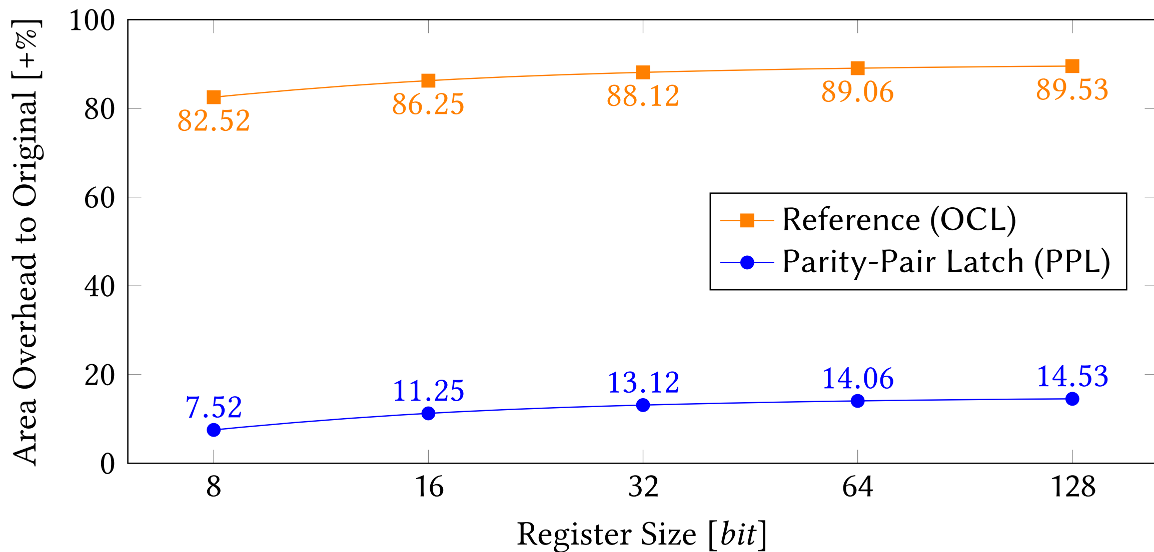

Figure 4.12.: Area Overhead - Parity Computation for a Single Register - Reference Implementation (OCL) and Parity-Pair Latch (PPL) (data from Table A.4).

Cell Library gates (Sec. 4.2.1, Eq. 4.4), and the *Parity-Pair Latch protection* (Sec. 4.2.2, Eq. 4.5). Absolute area values are provided in Table A.4 in the appendix.

The reference implementation has an area overhead of +82.52 % for a single 8-bit register. This overhead moderately increases with the register size due to the logarithmic growth of the parity tree. For 128 bits, +89.53 % of the original register size are required to compute the parity. The Parity-Pair Latch protection exhibits a significantly lower area overhead ranging from +7.52 % (8 bits) to +14.53 % (128 bits).

In summary, the application of Parity-Pair Latches to register parity computation significantly lowers the area overhead by 75 % independent of the register size. Thereby, the area necessary for the register parity computation is reduced from a almost doubling of the unprotected register area to less than 15 % additional area.

4.4.3. Single Event Upset Localization at Module Level

In presence of multiple registers, the modulo-2 address characteristic from Section 4.3 is used to localize Single Event Upsets as shown in Figure 4.1-d.

Area Overhead

To evaluate the overhead associated with the localization, the modulo-2 address characteristic is implemented and synthesized along with multiple parity protected registers, the characteristic register C, the syndrome computation as well as the derivation of the module wide fail signal. As previously, three configurations are considered, but from now on in the two-dimensional design space spanned by the size and the amount of registers.

Figure 4.13 depicts the normalized area for the previously used register sizes. For each register size, multiple registers are used to implement between 256 bits and 4096 bits in total. Consequently, between 32 and 512 8-bit registers are considered, while the number of 128-bit registers is between 2 and 32. For 8-bit registers, the area overhead associated with the detection and localization is between +99.1 % (256 total bits) and +97.68 % (4096 total bits), effectively doubling the area of the unprotected register. With growing register sizes, the area overhead reduces as the constant amount of total bits is implemented in fewer registers. Hence, area is saved by using a shorter characteristic at the cost of a decreasing localization resolution. The area overhead is

significantly reduced if Parity-Pair Latches are used. It ranges from +23.4 % to +22.6 % for 8-bit registers and further reduces for larger register sizes.

To quantify the particular overhead of the localization, the difference to the parity registers can be considered as follows. The characteristic of 512 parity bits is derived when 4096 bits are implemented using 8-bit registers. By subtracting the overhead for the 8-bit registers of +7.52 % from the total overhead of +22.6 % reported here, the localization overhead is +15.08 %. Similarly, 256 bits implemented in two 128-bit registers lead to a localization overhead of +1.17 % (+15.7 % − +14.53 %).

It is worth mentioning, that, if a constant amount of total bits is considered, whenever the localization overhead is high due to many small registers being used, the detection overhead is low. On the other hand, the use of fewer larger registers with a higher detection overhead implies a low localization overhead.

In summary, independent of the register organization, the detection and localization of Single Event Upsets has a relatively constant area overhead between one quarter and one sixth of the unprotected sequential portion of a module.

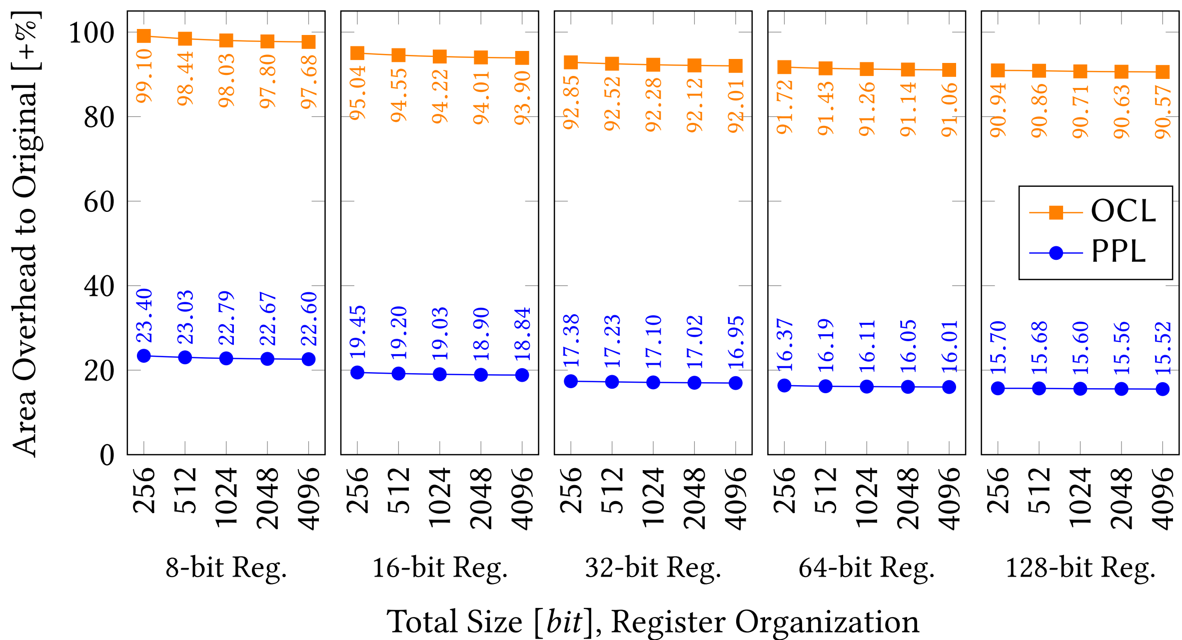

Figure 4.13.: Area Overhead - SEU Localization at Module Level - Reference Implementation (OCL) and Parity-Pair Latch (PPL) (data from Table A.5).

4.5. Summary

This chapter focused on the detection and localization of Single Event Upsets in registers contained in a module during clock gated phases. The presented two-tiered architecture provides dedicated solutions for the detection and localization implemented on different levels of abstraction.

The detection of Single Event Upset is implemented at gate level by deriving a parity checkbit for each individual register. Within a register, the first level of the parity tree is merged with the register latches into a new gate, the Parity-Pair Latch. Its implementation as a new standard cell is shown to be faster, to require less area, to exhibit a lowered peak and average power, as well as to result in a lower energy than the reference implementation. Consequently, the computation of register parities is shown to require a very low area effort.

The localization of Single Event Upsets across multiple registers is achieved at module level with the help of a checksum. The used modulo-2 address characteristic is able to detect and correctly localize all single errors. By using the optimal tree organization the amount of required gates as well as connections is significantly reduced and paves the way for an area efficient implementation.

By combining the detection at gate level with the localization at module level, an effective detection and localization of Single Event Upsets is facilitated, which can be implemented efficiently.

Chapter 5
Concurrent Online Correction of Single Event Upsets

The occurrence of soft errors is not limited to the clock gated phase. During operation, Single Event Upsets have the potential to result in additional failure modes due to an enabled clock. As in the previous section, the impairment of data stored in registers leads to data corruption, which potentially spreads to all data deduced from the affected register during successive computations. More severe is the corruption of the module's state itself, as the control flow is altered and thereby the sequence of instructions executed is changed.

The efficient detection and localization of Single Event Upsets during clock gated phases from the previous chapter is not limited to the clock gated phase and can also be applied during operation with a continuously enabled checksum computation. However, usually tighter requirements arise during operation that must be carefully considered during online Single Event Upset correction:

- Error Containment: With an enabled clock, all data stemming from the original upset location is corrupted. Thus, correction entails a complete recomputation.
- Localization Granularity: Correction modes more sophisticated than straight forward recomputation demand an error localization within the failing register.
- Correction Latency: The correction itself must be completed with a very low latency to reduce the impact of fault tolerance on operation performance.

This chapter is based on the online correction presented in [IW11a; IW11b], which copes with all these requirements by a self-contained architecture that provides an area, time, and power efficient online correction.

5.1. Concurrent Online Architecture

The concurrent online architecture builds upon the fundamentals of the previously described detection and localization. The architecture treats the registers of a module individually and allows for error detection, localization and correction. Therefore, in the following, a single register composed of n latches is considered (Figure 5.1-a).

The detection of transient errors is realized by protecting the register content with a checksum (Figure 5.1-b). The modulo-2 address characteristic discussed earlier is used and attached directly to the register (Block I). Thus, a sufficient localization granularity is provided in combination with an area efficient implementation. The register reference characteristic C is derived and stored in $log(n)$ additional latches upon the beginning of each clock cycle (Block II). Throughout the clock cycle, the recomputed characteristic C' is calculated concurrently from the register value and compared to the reference characteristic. The resulting syndrome S comprises the localization information, which as opposed to the non-concurrent detection now pinpoints to individual register bits. The syndrome is then compacted into a single bit *fail signal* by a simple OR tree and used to trigger a correction by recomputation. [1]

To implement an online correction that is faster than recomputation, the scheme is extended with a correction mechanism (Figure 5.1-c). In case of an error, the raised fail signal of a register is used to disable the clock of all other registers within the module which is beneficial in two ways. On the one hand, this local clock gating ensures that no new data is written to the affected register. Hence, the redundant information stored in the checksum, as well as its relationship to the register content is preserved. On the other hand, the error effect is contained in the register and can be corrected locally, within an extended timeframe procured by the local clock gating. To perform the actual correction, the register is composed of *n Bit-Flipping Latches (BFL)*, which are inherently able to invert their internal state (Block III). As the syndrome S directly derives the address of the affected bit, it is decoded and used to control the Bit-Flipping Latches to reconstruct the correct register state.

For all presented configurations no additional elements are inserted into the data path, thus no additional delay is introduced in the fault free case. If a single event upset occurs, either the global recomputation is triggered (Figure 5.1-b) or one additional clock cycle is used to correct the effect of the upset (Figure 5.1-c).

[1] The syndrome also allows fine-grained decisions, e.g. based on the significance of the affected bit.

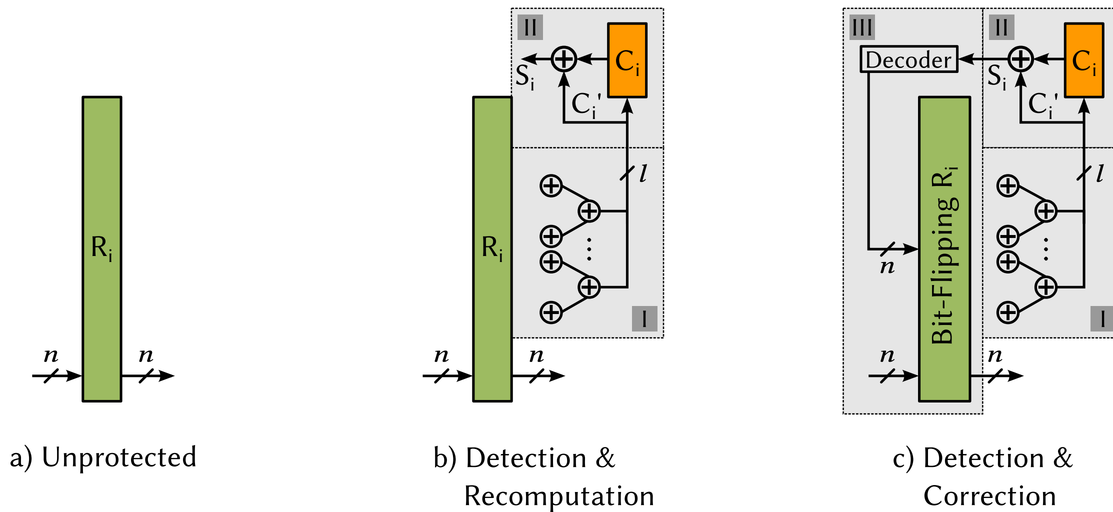

Figure 5.1.: Presented Concurrent Online Configurations.

Throughout this chapter, soft errors are confined to the occurrence of *Single Bit Upsets (SBU)* (single error assumption). The detection and correction in presence of multiple errors will be discussed in Chapter 6.

The remainder of this chapter starts with the Single Error Detection in Section 5.2. It briefly depicts the application of the modulo-2 address characteristic to individual registers (Block I), as well as the protection of the stored error condition in order to eliminate false detections (Block II). Subsequently, Section 5.3 details the extension to Online Correction with a focus on the Bit-Flipping Latch (Block III). Prior to the summary, both parts are experimentally evaluated in Section 5.4.

5.2. Single Error Detection (SED)

Upon the detection of errors during operation, a correction by recomputation requires many clock cycles during which a module is not available. Thus, the avoidance of non-essential corrections has the potential to reduce the performance impact associated with the time consuming restart of a computation. In the course of this section the base for such fine-grained correction decisions is laid by

- ▷ an improved localization granularity to spot errors within registers and
- ▷ the avoidance of false detections whenever the data path is correct.

5.2.1. Derivation of a Register Specific Error Condition

The modulo-2 address characteristic discussed in Section 4.3 has been shown to be an effective solution for the localization of SEU across multiple registers of a circuit. The need for an online detection and localization of SEUs within the registers can be fulfilled by deriving a characteristic for each individual register.

In contrast to the non-concurrent localization, the *register characteristic* C of a single register R with n bits is derived as $C = M \cdot R$. The characteristic is computed by attaching the characteristic tree directly to the register as shown in Figure 5.2 and stored in the l-bit reference characteristic register C. The syndrome S is derived by comparison of C and C' and compacted into the one bit syndrome fail signal *fail S* to indicate any discrepancy between the register content R and its characteristic C.

The implementation for each register profits from the low number of connections and gates enabled by the optimal characteristic tree organization introduced in Section 4.3.2. As a register is locally confined to a relatively small area in the final layout, only few routing resources are occupied and a small area overhead of the protected register is expected. The characteristic tree only requires exclusive OR standard cells and the associated area is identical to the non-concurrent localization (Eq. (4.12)). As all remaining infrastructure parts solely depend on the logarithmic characteristic, the area associated with the derivation of the error condition (EC) within a single register is low (Eq. (5.1)).

$$Area_{EC} = \underbrace{n \cdot A_{Latch}}_{\text{Register R}} + \underbrace{(2^{l+1} - 2l - 2) \cdot A_{XOR2}}_{\text{Characteristic Tree (Eq. (4.12))}} + \underbrace{l \cdot A_{Latch}}_{\text{Register C}} + \underbrace{l \cdot A_{XOR2}}_{\text{Syndrome S}} + \underbrace{(l-1) \cdot A_{OR2}}_{\text{fail S}} \tag{5.1}$$

The application of the modulo-2 address characteristic to individual registers provides the localization granularity necessary to identify single failing bits and will later allow to reconstruct the correct bit value at the erroneous location.

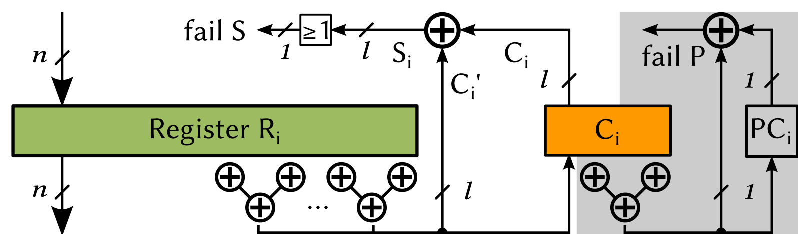

Figure 5.2.: Block I and Block II: Deriving and Protecting the Error Condition.

5.2.2. Protected Storage of the Error Condition

The modulo-2 address characteristic is able to detect and localize single errors in the protected register. The syndrome fail signal *fail S* will indicate any Single Event Upset in R. Unfortunately, a Single Upset in the stored reference characteristic C also leads to a raised syndrome fail signal (Figure 5.2). Hence, a false detection is triggered although the register content and thereby the data on the data path is unaltered and valid. To decide if a SEU affected the original register and thereby the data path or if it manifested in the stored reference characteristic, an additional protection of the error condition is necessary to distinguish both cases under a single error assumption.

This can be achieved by either protecting the register R or the characteristic C with an additional parity bit. The parity of the characteristic C is chosen due to a smaller area overhead resulting from the logarithmic relationship between the register size n and the characteristic size l (Eq. 4.6). The parity is then computed from the reference characteristic register C by a small XOR tree and stored in one additional latch PC as shown in the gray part of Figure 5.2. In the following, the difference of the reference parity P and the actual parity P' is called the parity fail signal *fail P*. Any Single Bit Upset in the register R, its characteristic C or the parity P thereof, will be visible in at least one of the two fail signals.

With two fail signals at hand, that indicate a discrepancy between either R and C (*fail S*), or C and P (*fail P*), the location can be derived for an error affecting the

- ▷ Original Register (R): The syndrome fail signal is '1' (as $C \neq C'$), the parity fail signal is '0' ($P = P'$). As the data on the data path is affected, a correction is triggered.

- ▷ Reference Characteristic (C): Both, the syndrome and the parity fail signal are '1' ($C \neq C'$ and $P \neq P'$). The error changed C, not the data path, no correction is needed.

- ▷ Characteristic Parity (P): The syndrome fail signal is '0' while the parity fail signal is '1' ($C = C'$ and $P \neq P'$). The reference parity P was altered. The data path is correct, no correction is needed.

Hence, the correction signal is defined as $correct = fail\,S \wedge \neg fail\,P$. The negation of the parity fail signal is efficiently implemented by merging with the XOR2 gate and

using a XNOR2 gate instead. The additional area to protect the error condition is small and grows logarithmically with the register size (Eq. (5.2)).

$$Area_{SED} = \underbrace{Area_{EC}}_{\text{Eq. (5.1)}} + \underbrace{(l-1) \cdot A_{XOR2}}_{\text{Parity Tree of C}} + \underbrace{A_{Latch}}_{\text{Register PC}} + \underbrace{A_{XNOR2}}_{\neg fail\ P} + \underbrace{A_{AND2}}_{correct} \tag{5.2}$$

Detection Capability: All single faults can be detected, correctly localized and corrected. Double faults can be detected, but not corrected. The detection of other multiple faults cannot be guaranteed. In general, the Hamming distance of the used code can be increased to allow the detection, localization and correction of multiple faults. [2]

Consequently, Single Error Detection (SED) is achieved, which completely avoids false detections. In case of a detection, detailed localization information within a register is supplied. Thus, fine-grained decisions are permitted for correction by recomputation.

5.3. Single Error Correction (SEC)

The previous section described how SEUs can be detected and localized while allowing more sophisticated correction decisions and reducing the amount of corrections by recomputation to a minimum. This section extends the scheme with a much faster online correction that directly utilizes the localization information to invert the affected bit while preserving the state of all other latches.

5.3.1. Rapid Correction by Bit-Flipping

To implement such a fast correction, only few building blocks are needed on top of the previously discussed detection. The localization of altered bits is already performed by the syndrome S which contains the address of the Single Bit Upset in binary form and is decoded by an 1-out-of-l decoder into a n-bit correction vector. By gating the syndrome with the *correct* signal in front of the decoder, the correction vector contains a single '1' if a correction is advisable and is the zero-vector otherwise. The

[2] The Extended Modulo-2 Characteristic for double error detection is discussed in Chapter 6.

area of the correction is hence given by Equation (5.3). The last summand denotes the area delta of the bitwise correction which will be discussed next.

$$Area_{SEC} = \underbrace{Area_{SED}}_{\text{Eq. (5.2)}} + \underbrace{l \cdot A_{AND2}}_{\text{Gate}} + \underbrace{l \cdot A_{INV2} + n \cdot (l-1) \cdot A_{NAND2}}_{\text{Binary Decoder}} + \underbrace{n \cdot (A_{BFL} - A_{Latch})}_{\text{Bit Flipping Overhead}} \tag{5.3}$$

5.3.1.1. The Bit-Flipping Latch

The actual correction of the erroneous bit is then performed by so-called *Bit-Flipping Latches (BFL)* as shown in Figure 5.1-c. Compared to other solutions for inverting the latch content, these have the advantage to go without additional gates or multiplexers in the data path that could compromise the timing behavior.

In general, a latch consists of two inverters (INV) and two transmission-gates (TG) (gray in Figure 5.3, with a direct connection between Q and TG5). It is controlled by a pair of control signals $\{L, \overline{L}\}$ which decides if a new value is latched from D (the latch is transparent) or if the internal state is preserved (the latch is blocking).

The *Bit-Flipping Latch (BFL)* is an extension, that contains an additional inverting feedback loop (TG4, inverter INV3 and TG3) (Figure 5.3). The reversion of the latched valued is controlled by an additional control signal pair $\{HI, \overline{HI}\}$ (Figure 5.3) as depicted in the following.

- ▷ *Hold Mode:* For $\{HI, \overline{HI}\} = \{1, 0\}$ the upper loop is used (TG2 and TG4 conducting, TG3 blocked), and the BFL behaves like a normal latch without compromising the timing behavior. In addition, the inverter INV3 is precharged by TG4. Hence, already holding the inverted value of Q ready at its output.

- ▷ *Invert Mode:* By inverting the additional control signal pair while the latch is blocking ($\{HI, \overline{HI}\} = \{0, 1\}$), both loops are blocked (TG2 and TG4 blocked, TG3 conducting) to avoid metastability of the inverting feedback loop. Simultaneously, the inverted value of Q is fed from INV3 to the inverter chain.
  As soon as the inversion of $\{HI, \overline{HI}\}$ is canceled in the next clock cycle, the non-inverting feedback loop stores the inverted value.

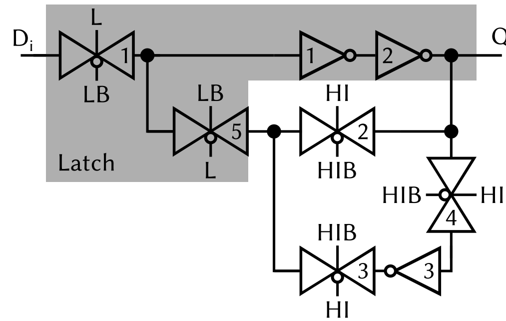

Figure 5.3.: Block III: Schematic of the Bit-Flipping Latch (BFL).

5.3.2. Timing Behavior of the Online Correction

With this effective correction mechanism implemented at bit-level, the course of events during the online correction implemented at register-level is now exemplified. Figure 5.4-a visualizes the timing of an unprotected register in a level-sensitive design with two non-overlapping clocks A and B (Figure 5.1-a). The timing of the presented correction scheme (Figure 5.1-c) is shown in Figure 5.4-b. A soft error hits the register in clock cycle clk_i at time t_1 and becomes visible at the register output at time t_2. The online correction detects the changed value at time t_3. The raising *correct* signal gates the clock signal B, which controls all predecessing and subsequent registers in the level-sensitive design style and triggers the correction, which is completed at time t_4. The falling *correct* signal at t_5 in clock cycle clk_i' indicates the successful correction.

5.4. Experimental Evaluation

The Online Correction of Single Event Upsets is now evaluated. The description of the experimental setup is followed by the results for the Single Error Detection relying on correction by recomputation. The review of the Single Error Correction first focuses on the Bit-Flipping Latch serving as the basic building block before evaluating the protection of complete registers. The used tools are particularized in Section A.1, while Section A.3.2 supplies tabulated results where appropriate.

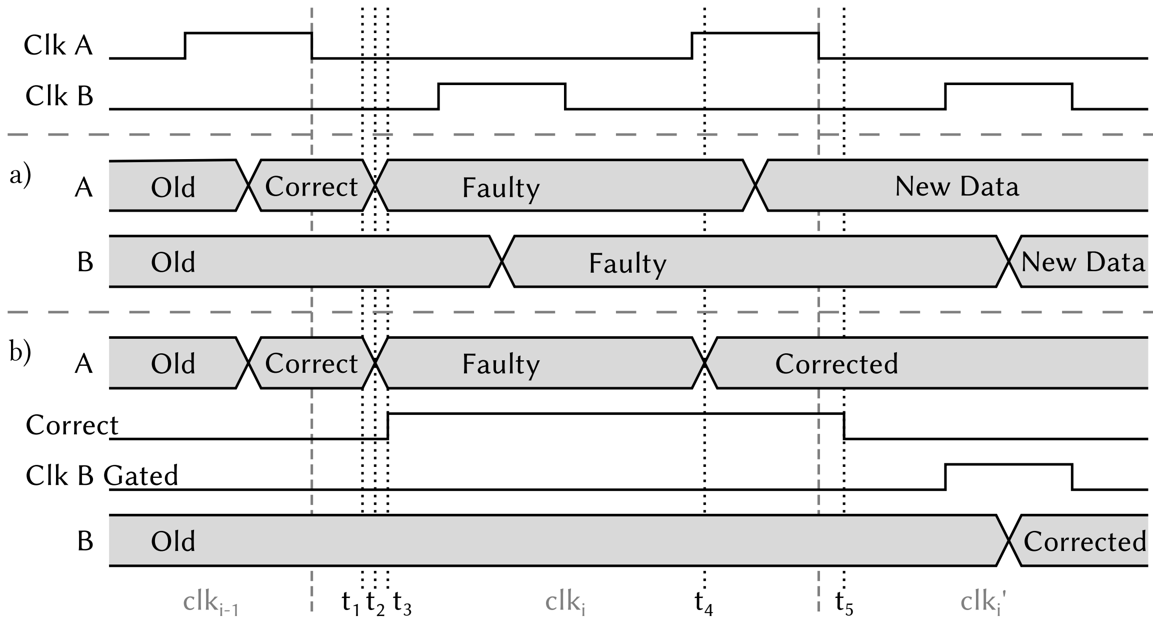

Figure 5.4.: Timing Behavior in Presence of Soft Errors: a) Unprotected Register (Figure 5.1-a); b) Protected Register with Correction (Figure 5.1-c).

5.4.1. Experimental Setup

The Online Single Error Detection from Section 5.2 is implemented as a gate level netlist in VHDL, synthesized for the 45 nm Nangate *Open Cell Library (OCL)* [Nan11] and compared to classical *Duplication With Comparison (DWC)* in terms of area.

For the Rapid Correction by Bit-Flipping from Section 5.3 first the Bit-Flipping Latch from Section 5.3.1 is implemented as a new standard cell. As previously, the cell is designed according to the design rules and electrical rules of the FreePDK *Process Design Kit* [SCW+07] in order to be compatible with the *Open Cell Library (OCL)* cells. Thereafter, consistency with the schematic is checked and confirmed during *Layout-vs-Schematic (LVS)* and a transistor netlist containing all parasitic layout effects is obtained during *Physical Extraction (PEX)*. The netlists of the Bit-Flipping Latch and the standard latch from the *Open Cell Library* are then simulated at the analog level using SPICE [NP73]. A rising and a falling edge with a slew rate of 22 V/ns are employed to characterize the timing behavior and power consumption. An inverter (INV_X1) is consistently used as a load at the output and 10 % as well as 90 % of the nominal voltage serve as the used trip points.

Subsequent to *Library Characterization* the standard cell is added to a new library and used in conjunction with cells from the *Open Cell Library* to extend the VHDL netlist used for Single Error Detection with the rapid Single Error Correction. Last, the area overhead involved in protecting single registers is compared to two classical approaches across different register sizes, namely *Triple Modular Redundancy (TMR)* and bitwise *Fault Tolerance (FT)* comparable to RAZOR [EKD+03] and GRAAL [Nic07] (see Sec. 3.1.2). Finally, the Timing Behavior of the Online Correction is validated through a fault injection experiment.

5.4.2. Single Error Detection (SED)

The Single Error Detection relies on the derivation of a logarithmic checksum for each individual register. The used modulo-2 address characteristic was already shown to possess a lightweight implementation in Section 4.3. Therefore, the area overhead of its direct application to single registers will be evaluated.

Area Overhead

In the following, an unprotected register serves as a baseline to allow the quantification of the normalized area overhead across different register sizes. It is implemented using the high enable latches (DLH_X1) from the Open Cell Library, where each latch has a cell height of 1.4 μm, a width of 1.9 μm, and an area of 2.66 μm^2.

The bit-wise detection by *Duplication With Comparison (DWC)* is implemented for comparison and better classification of the area efficiency. In DWC, for each register bit, two OCL DLH_X1 latches and a XOR2_X1 exclusive OR are used to derive the error condition of the bit. These bit error conditions are then aggregated into the register error condition by an OR tree (OR2_X1). The area overhead of DWC is depicted in Figure 5.5; it is approximately +190 % and does not depend on the register size.

The Single Error Detection extends the n-bit register block (using DLH_X1 cells) with the characteristic computation, the additional register to store C and the comparator. An OR-tree aggregates the computed syndrome S into the *syndrome fail* signal. Protecting the reference characteristic C with an additional parity determines if a soft error affected R or C. The area overhead of the detection actually decreases with growing register sizes as all used components depend logarithmically on the register size (Eq. (4.6), Eq. (5.2)). Compared to the area of the unprotected register,

the presented detection introduces an area overhead reaching from +177.12 % for a 7-bit register down to +94.31 % for a 255-bit register.

In summary, for small registers, the presented online detection already has a smaller area overhead which further reduces with growing register sizes. For large registers a significantly lower area overhead is achieved by turning away from bitwise redundancy in favor of a logarithmic checksum.

5.4.3. Single Error Correction (SEC)

Implementing the rapid correction on top of the online detection requires only few extensions. A syndrome decoder is added to control the bit-level correction which is implemented by replacing the register latches with Bit-Flipping Latches. As this substitution directly affects the data path independent of the occurrence of Single Event Upsets, the properties of the designed Bit-Flipping Latch standard cell will be evaluated first. In succession, the protection of complete registers is evaluated along with the timing behavior of the online correction.

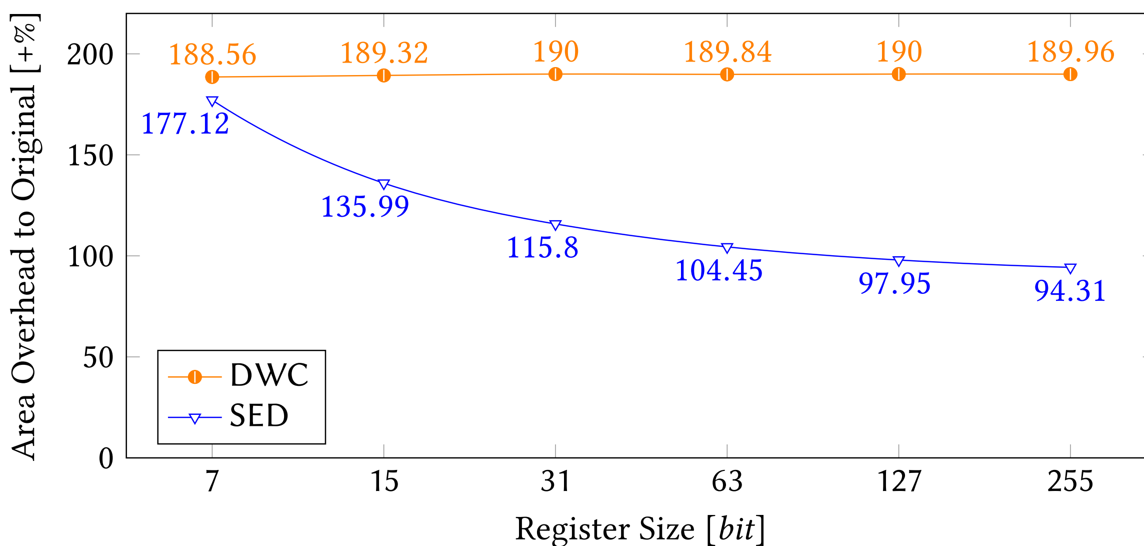

Figure 5.5.: Area Overhead - Single Error Detection (SED) - Single Register (data from Table A.6).

5.4.3.1. Bit-Flipping Latch Standard Cell

The Bit-Flipping Latch is implemented as a new standard cell using a full custom design style. This building block fundamental to the online correction will be evaluated with respect to three metrics. The cell area to quantify the correction area overhead; the timing behavior to except any negative impact during error free operation; as well as the power consumption and energy to depict savings over the replaced latches.

Standard Cell Area

Figure 5.6 shows the layout of the Bit-Flipping Latch standard cell. The order of the single gates from the schematic (Figure 5.3) from left to right is: TG1, TG5, INV1, INV2, TG2, TG4, TG3, INV3. The left part of the cell implements a latch (TG1, TG5, INV1, INV2), while the right part contains the additional inverting feedback loop. The DLH_X1 latch from the Open Cell Library is used for comparison which has an area of 2.66 μm^2. The Bit-Flipping Latch uses the same cell height of 1.4 μm to maintain compatibility. Together with the width of 2.28 μm, the area is 3.192 μm^2. Hence, compared to the OCL DLH_X1 latch, the additional area for implementing the inverting feedback in the Bit-Flipping Latch is as low as +20 %. [3]

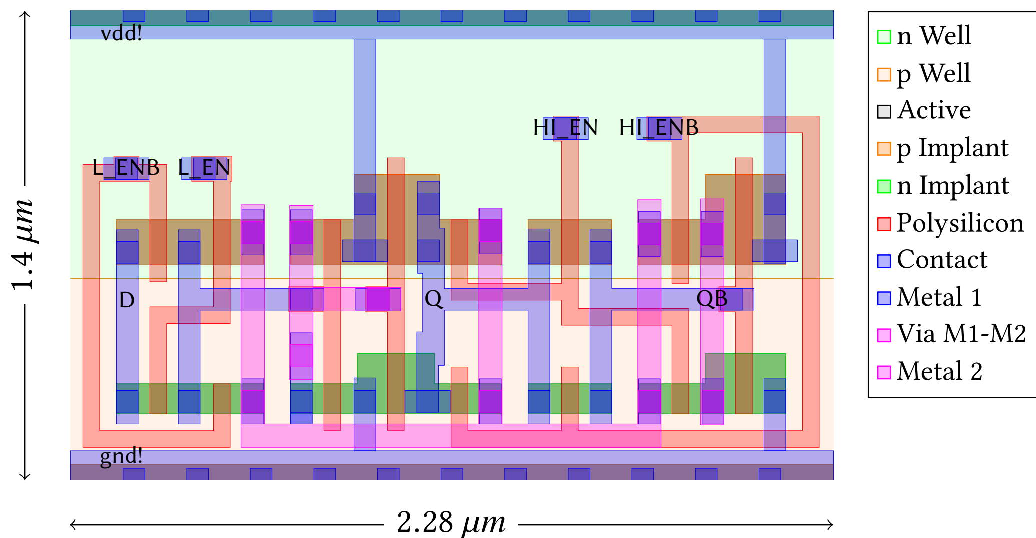

Figure 5.6.: Layout of the Bit-Flipping Latch Standard Cell BFLATCH_X1.

[3] In [IW11a] an older version of the Open Cell Library was used, which led to an area overhead of +9 % due to a larger reference area of 2.926 μm^2 instead of 2.66 μm^2 for the DLH_X1 latch.

Timing Behavior of the Bit-Flipping Latch

To identify any negative bias of the correction scheme on the timing behavior in the fault free case, the delay of a high enable latch from the OCL and of the Bit-Flipping Latch is compared using SPICE. Figure 5.7 plots the voltage at the input D and the output Q of both latches over the simulation time of $500\,ns$.

The output of the OCL latch (orange dashed) reaches $0.11\,V$ after $81.93\,ps$ for a falling edge. For the rising edge $0.99\,V$ are reached after $76.1\,ps$. [4]

The Bit-Flipping Latch (blue dotted) has a delay of just $70.59\,ps$ for the falling edge. For the rising edge, the delay of $77.28\,ps$ is only marginally larger as for the reference. Thus, the Bit-Flipping Latch is faster than the standard latch from the library, which is explained by a careful scaling of the used transistors.

Consequently, the application of Bit-Flipping Latches has no negative influence on the timing behavior of a circuit.

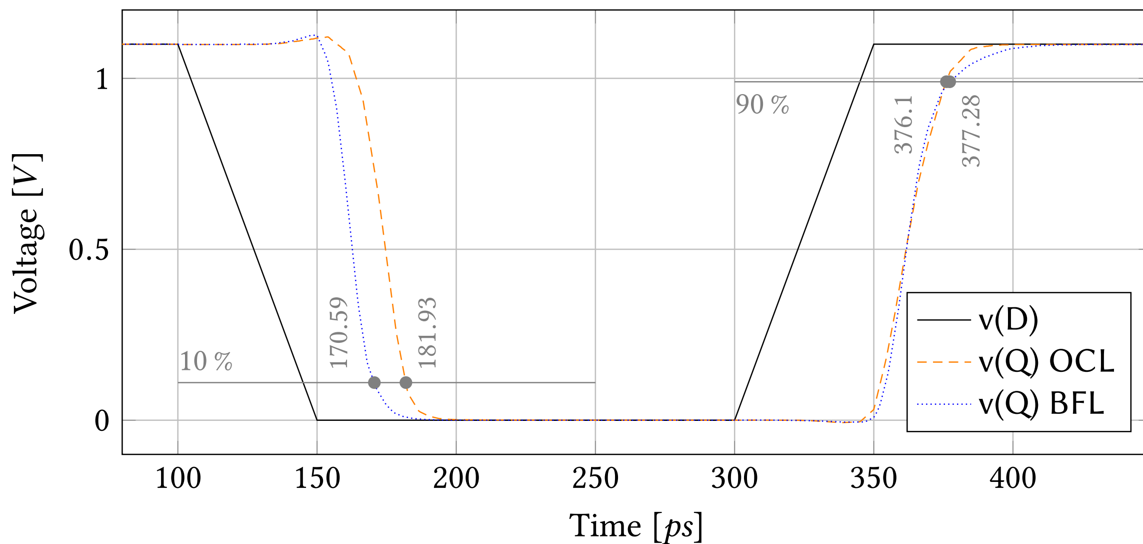

Figure 5.7.: Timing Behavior of the OCL Low Enable Latch (DLH_X1) and the Bit-Flipping Latch (BFLATCH_X1): D-to-Q Delay.

[4] In comparison to Chapter 4 the DLH_X1 latch is faster due to the reduced load of a single inverter instead of an inverter and an exclusive OR in the PPL reference implementation.

Power Consumption and Energy

The power consumption of the reference latch and the Bit-Flipping Latch during the analog simulation are depicted in Figure 5.8.

For the DLH_X1 latch, the instantaneous power consumption for the falling edge is significantly lower than for the rising edge with a peak power consumption of 80.91 μW. The power consumption of the Bit-Flipping Latch is more balanced and a slight increase of the peak power to 83.51 μW is observed. The average power consumed during simulation by the DLH_X1 latch is 10.88 μW. As the Bit-Flipping Latch has a shorter delay for the falling edge, its power consumption is higher and shorter. Consequently, the average power of the Bit-Flipping Latch is lower and amounts only 8.37 μW or 76.93 % of the average power for the reference latch. The energy (area under the curves in Figure 5.8) is reduced by the same fraction due to the identical simulation time.

In summary, the Bit-Flipping Latch has a comparable peak power consumption as the superseded latch, while the average power consumption and energy are both reduced by almost a quarter.

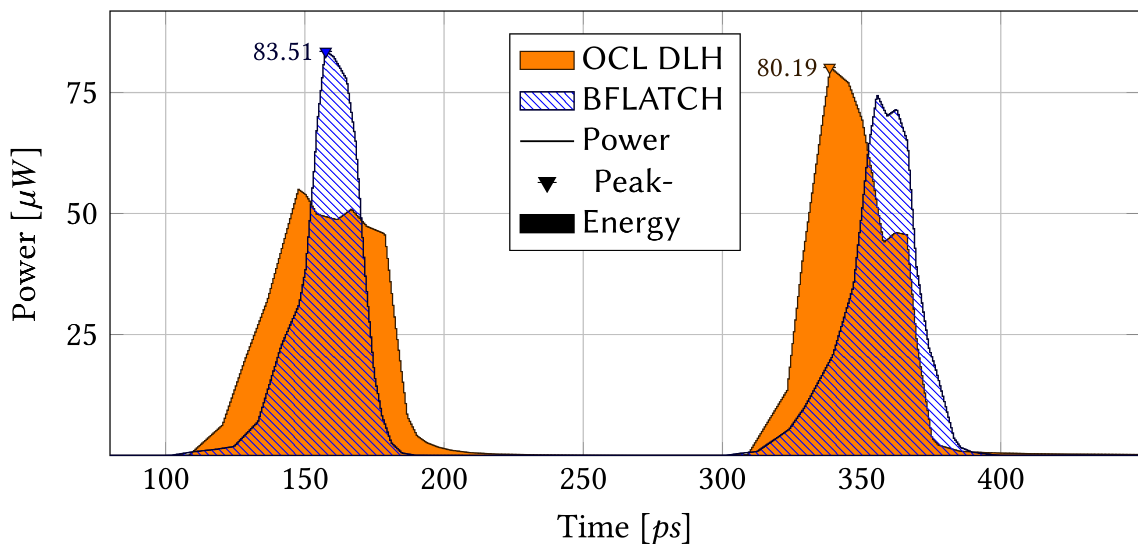

Figure 5.8.: Power and Energy of the OCL Low Enable Latch (DLH_X1) and the Bit-Flipping Latch (BFLATCH_X1).

5.4.3.2. Correction within a Single Register

Now, the application of Single Error Correction to complete registers is evaluated which makes use of the efficient bit-level correction enabled by Bit-Flipping Latches.

Area Overhead

For comparison, *Triple Modular Redundancy (TMR)* and bitwise *Fault Tolerance (FT)* similar to RAZOR [EKD+03] and GRAAL [Nic07] (see Sec. 3.1.2) are considered. Triple Modular Redundancy is implemented by three DLH_X1 latches and a majority voter (see Figure 3.2-b). For the voter, an area optimized implementation with three NAND2_X1 and one OR2_X1 gate is used instead of the straight forward use of three AND2_X1 and two OR2_X1 gates to allow an unbiased examination. The area overhead of TMR depicted in Figure 5.9 is constant across all register sizes and amounts +330 %. Bitwise Fault Tolerance consists of bit slices composed of a DLH_X1 latch, a shadow DFF_X1 flip-flop, a XOR_X1 exclusive OR, and a MUX2_X1 multiplexer to restore the value from the shadow flip-flop. All bit slice fail signals are aggregated into a global register fail signal by an OR tree (OR2_X1) (also optimized for area during synthesis) to restore the complete register upon failure of at least one bit. The area overhead of bitwise FT is +328.57 % for a 7-bit register and slowly grows with the register size.

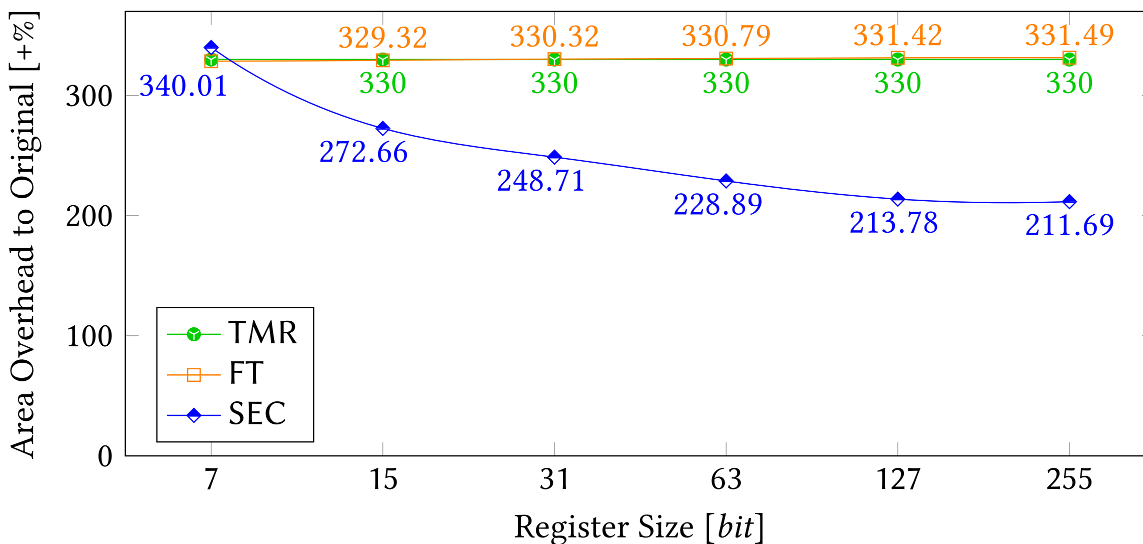

Figure 5.9.: Area Overhead - Single Error Correction (SEC) - Single Register (data from Table A.7).

The presented online *Single Error Correction (SEC)* is implemented according to Figure 5.1-c and includes all earlier discussed building blocks to compute, to store, and to protect the characteristic. The syndrome S is gated with the *correct* signal, decoded and used to control the Bit-Flipping Latches superseding the register latches. The Single Error Correction has an area overhead of +340 % for a 7-bit register, which is higher than TMR or FT. However, with growing register sizes, the logarithmic checksum can play to its strength and the overhead is significantly reduced. For a 127-bit register the area overhead is only +213.78 %.

Timing Behavior of the Online Correction

To determine the *Time Vulnerability Factor (TVF)* of the presented scheme, soft errors are injected into an 8-bit register protected with the correction from Section 5.3. The used clock period was 4 ns with a 25 % high phase (as depicted in Figure 5.4). During the high phase of the clock the latches are transparent; soft errors hitting the latch can lead to glitches, but cannot permanently alter the sequential state. During the low phase, the latches are locked and vulnerable to Single Event Upsets. A series of simulations was performed, where a soft error is injected into a randomly chosen latch by forcing the output of the first feedback inverter to its opposite value (injection time t_1). A testbench then recorded the following time points corresponding to t_2 till t_5 in the previous section: Visibility at the register output, raising correct signal indicating the upset, visibility of the corrected value at the output and a falling correct signal. The experiments were conducted for the whole low phase of 3 ns, while the injection time point was moved forward by 100 ps for every experiment. The results in Table 5.1 show, that all SEUs during the low phase are detected and corrected.

Table 5.1.: Time Vulnerability: 8-bit Register with Single Error Correction (SEC).

Injection Time [ps]	Visible at output [ps]	Detected (*correct* rising) [ps]	Corrected at output [ps]	End (*correct* falling) [ps]
0 (bit 4)	100	300	3200	4300
100 (bit 6)	200	300	3200	4300
200 (bit 6)	300	400	3200	4300
...	...	...	...	...
2600 (bit 0)	2700	2800	3200	4300
2700 (bit 1)	2800	2900	3200	4300
2800 (bit 5)	2900	3000	3200	4300

Example In Figure 5.4-b the line starting with "200 (bit 6)" from Figure 5.1 represents the experiment where an error is injected in bit 6 at time $t_1 = 200\,ps$, visible at the output at $t_2 = 300\,ps$, and detected at $t_3 = 400\,ps$. It is corrected at $t_4 = 3200\,ps$ and the falling *correct* signal indicates the successful correction at $t_5 = 4300\,ps$.

5.5. Summary

Resilience to soft errors is nowadays considered a necessity. However, resilient designs require additional circuitry or clock cycles to detect and correct soft errors. Hence, the application of fault tolerance has to trade off design robustness against design quality and cost (area, performance and power).

The presented Soft Error Detection conquers the area overhead associated with a bitwise implementation by deriving a register-wide checksum. The employed modulo-2 address characteristic is implemented as a tree with optimal organization and compacts the register content into a logarithmic error condition. The protected storage of the error condition completely avoids false detections and upon detection, the Single Bit Upset is correctly localized. Thereby allowing to trade off the necessity of correction by recomputation against the associated performance penalties at a higher abstraction level. During evaluation, the Soft Error Detection scheme is shown to not sacrifice performance of the data path and incorporate a low area overhead.

The presented Soft Error Correction overcomes the performance penalties associated with correction by recomputation such as increased computational latency and throughput degradation. Correction in one additional clock cycle is enabled by confining the necessary inversion of the affected register bit to a newly designed standard cell: The Bit-Flipping Latch. The cell is shown to combine no performance degradation during fault free operation with a reduced power consumption and a low hardware overhead. Rapid correction is rendered possible by exploiting the localization information procured during detection to control the bit-inversion inherent to the Bit-Flipping Latches. Overall, the Soft Error Correction scheme is shown to incorporate a lower area overhead than bitwise fault tolerance for widely used register sizes while limiting the performance impact of correction to a minimum.

Chapter 6
Fault Tolerance in Presence of Multiple Bit Upsets

Radiation induced soft errors originate from radiation events that are random in time and location. Most single events result in Single Bit Upsets (SBU) that are perfectly handled by the online correction discussed in the last chapter. With a lower probability, multiple bits may be affected concurrently by a single event with a sufficiently high energy. [1] For multiple upsets it is, independent of the root cause, important to differentiate how many bits are affected within a register block (the protected register and its fault tolerance infrastructure) at a time.

A *Multiple Cell Upset (MCU)* affects multiple bits of a module but each register block is struck by at most one upset. As the previously discussed fault tolerance scheme allows each register to correct a single error, the upsets can be corrected simultaneously. Hence, Multiple Cell Upsets are covered without additional effort.

In case of a *Multiple Bit Upset (MBU)* at least two bits within a single register block are affected. [2] As the online correction from the last chapter was designed under a single error assumption, amendments are needed to cope with multiple errors which will be discussed in the course of this chapter.

In theory, any arbitrary number of upsets affecting a register can be corrected under a sufficiently high minimum Hamming distance. As the probability of multiple bit upsets decreases with their multiplicity and as the cost associated with fault tolerance raises with the Hamming distance, it is mandatory to carefully trade off correction capabilities against their cost in terms of area overhead.

[1] Theoretically, multiple independent events may also occur simultaneously in very rare cases.

[2] The terms MCU and MBU are used in a generalized form that covers all sequential elements within a register and its protection in contrast to the strict definition per register from Section 2.2.1.

This chapter depicts the amendments to handle Multiple Bit Upsets and exemplarily extends the online correction from Chapter 5 to double errors. The remainder of this chapter starts with the analysis of the shortcomings of single error correction in presence of double errors. It is followed by the improved architecture in Section 6.2 and the employed extended characteristic in Section 6.3. Last, the architecture is experimentally evaluated in Section 6.4 before concluding with a short summary.

6.1. Preliminary Error Multiplicity Considerations

The online correction from the previous chapter employs two fail signals (syndrome fail *fail S* and parity fail *fail P*). Thereby, an effective distinction of the four error scenarios possible under a single error assumption is enabled as shown in the upper three dotted parts of Table 6.1. If double errors are considered in addition, the amount of possible error location combinations increases to 9 and two fail signals are no longer sufficient for their differentiation. As a consequence, false corrections cannot be prevented in three out of the five additional cases (the lower three dotted blocks with false detections highlighted in gray) as follows.

- ▹ For two errors in register R at addresses a_1 and a_2, the correction is performed at the wrong location $a_1 \oplus a_2$ as a single error location is assumed.
- ▹ For two errors in the characteristic register C, a false correction is not hindered by the parity fail signal *fail P*, as the actual characteristic parity collides with the stored reference characteristic parity for any even number of errors in C.
- ▹ For one error in the characteristic register C and one in the characteristic parity register PC, the parity fail signal *fail P* is zero as both, the actual and the reference parity are altered simultaneously.

Note that, for all of the cases discussed above that result in false corrections, the error multiplicity in register R is even. While, for the single case where a single error affects register R and a correction is desired, the error multiplicity of R is odd. Hence, to trigger only the desired correction, additional information needs to be provided by the fault tolerance architecture as discussed in the following.

Error Multiplicity	Error Location				Fail Signal			Decision	
	R	*C*	*PC*	*PR*	*S*	*P*	*E*	*correct*	*correct+*
None	0	0	0	0	0	0	0	0	0
Single	1	0	0	0	1	0	1	1	1
	0	1	0	0	1	1	0	0	0
	0	0	1	0	0	1	0	0	0
	0	0	0	1	0	0	1	na	0
Double	2	0	0	0	1	0	0	1	0
	0	2	0	0	1	0	0	1	0
	1	1	0	0	1	1	1	0	0
	1	0	1	0	1	1	1	0	0
	0	1	1	0	1	0	0	1	0
	1	0	0	1	1	0	0	na	0
	0	1	0	1	1	1	1	na	0
	0	0	1	1	0	1	1	na	0

Table 6.1.: Double Error Locations - Single Error Correction (SEC, dotted part, Section 5.2) and Single Error Correction, Double Error Detection (SECDED).

6.2. Online Architecture for Double Errors

To eliminate false detections in presence of double errors, the fault tolerance from Chapter 5 is extended. The online detection and correction shown in the dotted area in Figure 6.1 uses the l-bit modulo-2 address characteristic C as the error condition. In order to increase the Hamming distance of the error condition, the additional *register parity PR* is used. It can easily be computed by an exclusive OR tree and is stored in an additional latch *PR*. Similar to the protected storage of the reference characteristic, the reference register parity is compared to the actual register parity *PR'* and the extended fail signal *fail E* is derived.

The additional columns for the register *PR* and the fail signal *fail E* in Table 6.1 show that, for the above mentioned 9 cases, the extended fail signal *fail E* is one only for an error multiplicity of 1 where a correction is possible. As upsets cannot only affect the original register R, the reference characteristic register C and the characteristic parity *PC* but also the newly derived register parity *PR*, the four remaining cases where *PR* is directly altered by a Single Bit Upset are shown in Table 6.1.

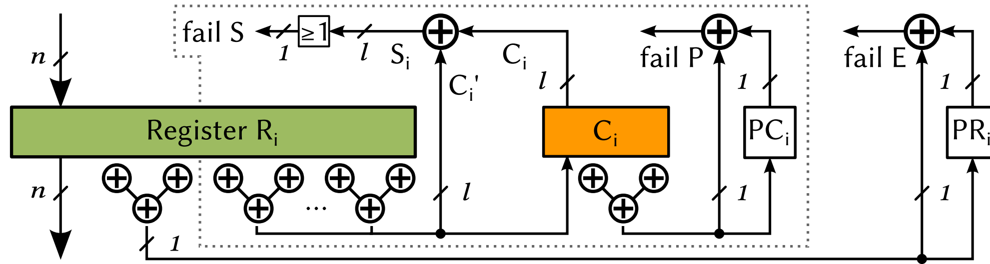

Figure 6.1.: Block I and Block II: Deriving and Protecting the Extended Error Condition in Presence of Double Errors.

With the addition of *fail E*, all cases can be distinguished and a correction is only performed when a single error affects the data path by defining the *extended correction signal* as *correct+* = *fail S* $\wedge \neg$*fail P* $\wedge$ *fail E*.

Also note the close relationship to the additional parity bit of an extended Hamming code (see Eq. 2.6 in Sec. 2.2.3.1) or the extension of error detecting memory refreshment by appending a constant 1 to every address (see Sec. 3.1.1.2).

6.3. Optimal Extended Characteristic Computation

The *extended modulo-2 address characteristic C+* combines the register parity *PR* and the register characteristic *C* into a $l+1$ bit wide error condition defined as follows.

Definition 6.3.1 (Extended Modulo-2 Address Characteristic) *Let $\vec{R}$ be a n-bit wide register with addresses $A = \{n, \ldots, 1\}$ and let $A_1 := a \in A \mid \vec{R}[a] = 1$ denote the set of all addresses containing a logic 1. Then, the extended modulo-2 address characteristic is defined as*

$$\vec{C}+ = (\bigoplus_{a \in A} \vec{R}[a] \, , \bigoplus_{a \in A_1} a) \quad .$$

In the following, the characteristic computation from Section 4.3.2 is augmented with the register parity computation. Thereby, the hardware overhead is reduced as logic sharing between the parity and the characteristic computation can be exploited.

The optimal organization of the characteristic computation was already discussed in detail in Section 4.3.2. To recapitulate: The characteristic tree derives the l characteristic bits $\vec{C} = (c_{l-1}, \ldots, c_0)$ in l tree levels as shown in Figure 6.2.

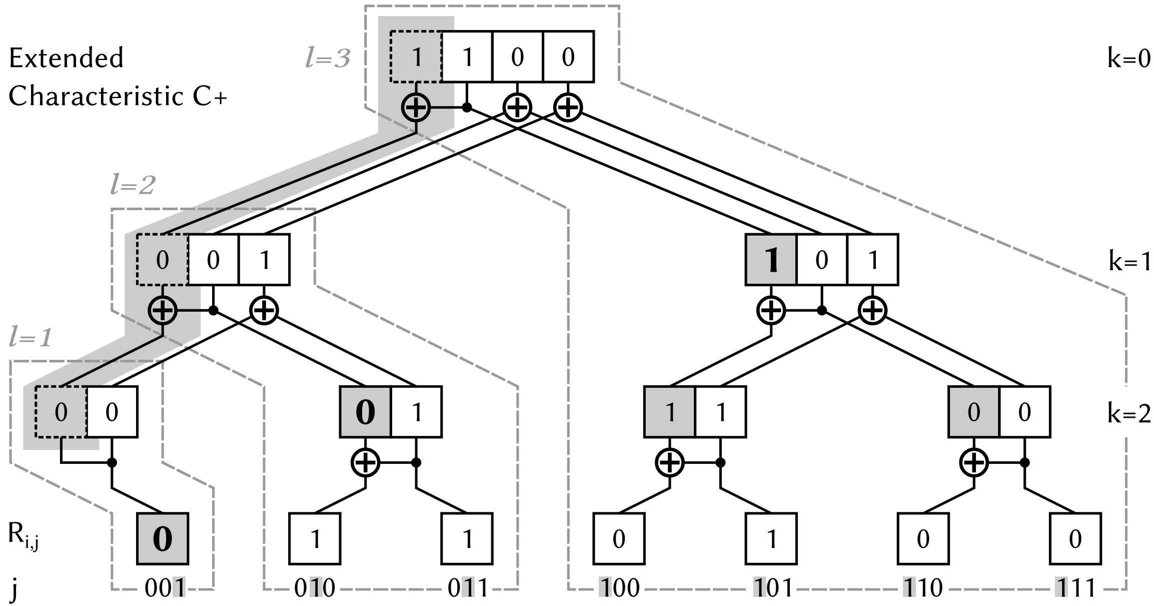

Figure 6.2.: Optimal Extended Characteristic Tree Organization.

In level $k \in (0, \ldots, l-1)$, exactly $(l-k)$ characteristic bits are derived, where the $(l-k-1)$ least significant characteristic bits $(c_{k,i,l-k-2}, \ldots, c_{k,i,0})$ (with white background in Figure 6.2) are computed from the characteristics of the predecessors.

The most significant bit $c_{k,i,l-k-1}$ (with gray background in Figure 6.2) corresponds to the most significant address bit $(l-k-1)$ that distinguishes the two predecessors and is known to be 0 respectively 1. It follows from Formula (4.10):

$$c_{k,i,l-1} = p_{k,2\cdot i+1} \,.$$

Hence, the most significant bit $c_{k,1,l-1}$ at level $k \in (1, \ldots, l)$ is equivalent to the parity of all (2^{l-k}) leaf nodes with a 1 at address bit $(l-k)$ (with bold text in Figure 6.2). Thus, the register parity *PR* is derived as the parity of these l sub parity bits by $(l-1)$ additional exclusive OR gates (contained in the gray highlighted slice at the left of Figure 6.2), which is significantly less than the $(n-1)$ exclusive OR gates for implementing a complete register parity tree.

In general, to dilate the extended characteristic from $l-1$ to l characteristic bits, one exclusive OR gate is required in addition to the sole characteristic tree: $2^l - 1$. The total amount of exclusive OR gates for implementing an extended characteristic of size l is calculated by summation over all levels. The area of the extended characteristic is

obtained by multiplication with the area of the used two-input exclusive OR.

$$Area_{Char+} = \sum_{2 \leq j \leq l} (2^j - 1) \cdot A_{XOR2} = (2^{l+1} - l - 3) \cdot A_{XOR2} \tag{6.1}$$

Example In Section 4.3.2, deriving the 3-bit characteristic of a 7-bit register required 8 XOR2 gates (Formula 4.12 with $N = 7, l = 3$). Here, just two additional two-input exclusive OR gates are required to compute the extended characteristic $C+$:

$$\sum_{2 \leq j \leq 3} (2^j - 1) = 2^4 - 3 - 3 = 16 - 3 - 3 = 10\,.$$

6.4. Experimental Evaluation

The extension of fault tolerance to cope with double bit upsets in single register blocks is now evaluated with respect to the involved area overhead. The employed tools and tabulated results are depicted in Section A.1, respectively Section A.3.3.

6.4.1. Experimental Setup

The Single Error Detection (SED) and the Single Error Correction (SEC) architecture from Section 5 are complemented with the parity fail signal as shown in Figure 6.1 and the correction signal is updated to implement the extended correction signal *correct+* according to Section 6.2. The optimal extended characteristic computation is used and implemented by augmenting the characteristic tree with the register parity computation as described in Section 6.3. The resulting architectures support Single and Double Error Detection (DED) as well as Single Error Correction Double Error Detection (SECDED) and are synthesized using the Nangate Open Cell Library and the additional library containing the Bit-Flipping Latch.

6.4.2. Area Overhead

The area overhead of the two newly derived architectures over an unprotected register is now evaluated and compared to the architectures from Section 5 and commonly used techniques which rely on a bitwise implementation.

The results for Single and Double Error Detection (DED) are shown in Figure 6.3. In comparison to sole Single Error Detection, the area requirements are higher due to the introduced constant area associated with the storage of the reference register parity and the derivation of the extended correction signal. With growing register size, the area overhead reduces logarithmically as the impact of amendments with constant area reduces. More importantly, the extended characteristic computation still grows logarithmically with register size. For a register size of 15 bits the DED architecture has an area overhead of +159.32 %, which is already lower than bitwise Duplication with Comparison (DWC). For 127-bit registers, the overhead is only +102.13 %, which is almost negligible compared to the SED architecture and just merely higher than register duplication (without comparison).

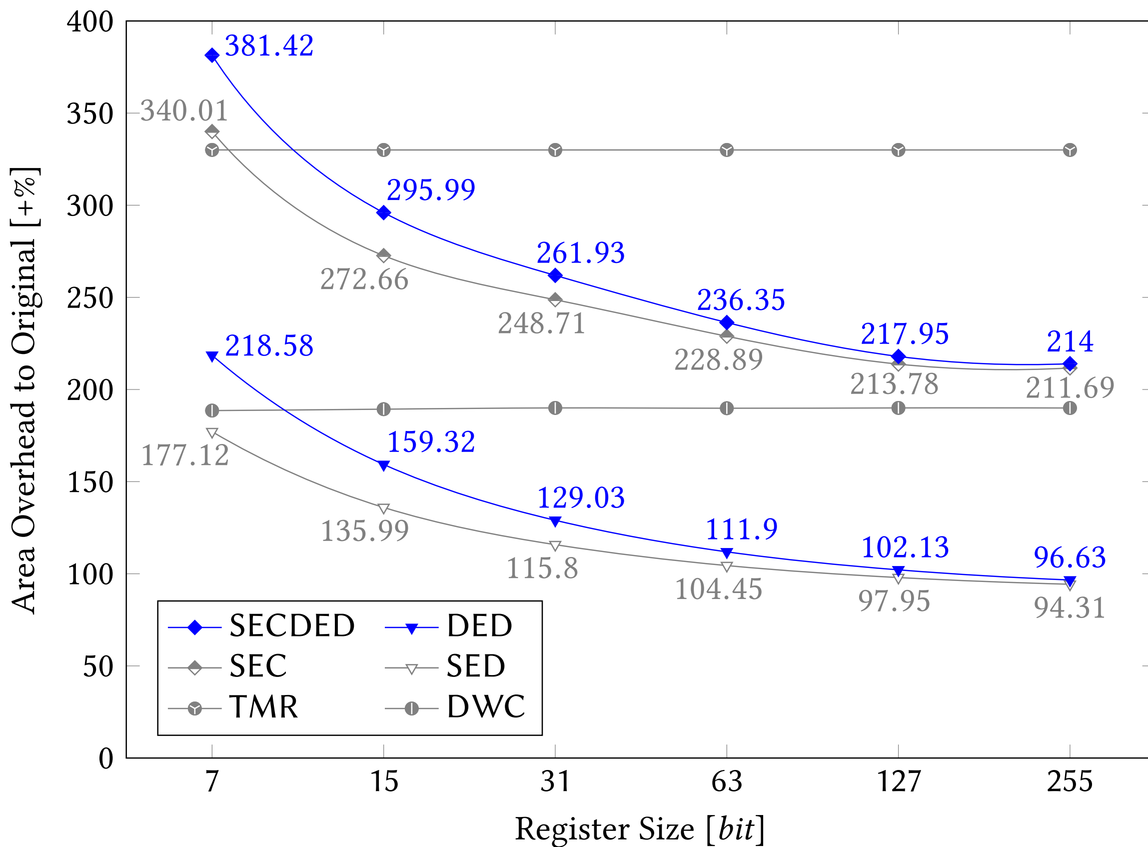

Figure 6.3.: Area Overhead - Single and Double Error Detection (DED), Single Error Correction Double Error Detection (SECDED) - Single Register (data from Table A.8).

Single Error Correction and Double Error Detection (SECDED) involves an elevated area overhead as depicted in Figure 6.3. It ranges from +295.99, % for a 15-bit register to +217.95 % for a 127-bit register, which is also only slightly higher than Correction without Double Error Detection. As for the DED architecture, the advantages of a register wide checksum start to pay off for register sizes of at least 15 bits, where the area overhead begins to fall below the overhead associated with bitwise Fault Tolerance (DWC with restore) or Triple Modular Redundancy (TMR).

6.5. Summary

The online protection from the last chapter was thoroughly analyzed with respect to its behavior in presence of multiple errors. The analysis showed, that double bit upsets or the coexistence of two single bit upsets may result in false corrections.

These false corrections were targeted by augmenting the *error condition* with an additional *register parity* bit. The newly derived *extended error condition* is shown to successfully avoid false detections for an error multiplicity of up to two. Thus, corrections are only cleared when the protected main register is reconfirmed to be affected by a Single Bit Upset that is correctable by the Bit-Flipping architecture.

The register parity computation is implemented by merging with the register characteristic computation into the *Extended Characteristic*. The Extended Characteristic is shown to possess an optimal combinational tree organization, which exhibits a very low additional area overhead of just $(l - 1)$ exclusive OR gates due to the exploitation of logic sharing between both trees.

The two architectures from the previous chapter were updated with the extended protection and thereby support *Double Error Detection (DED)* respectively *Single Error Correction Double Error Detection (SECDED)*. The area overhead after synthesis across various register sizes is shown to be lower than the overhead of classical approaches like Duplication with Comparison (DWC) or Triple Modular Redundancy (TMR) for registers with 15 or more bits. The area overhead is also shown to grow logarithmically with register size due to the use of a register wide checksum instead of bitwise redundancy.

Chapter 7
Area Efficient Characteristic Computation

The previous chapters showed, how fault tolerance in random logic is achieved by applying a logarithmic checksum to individual registers. The depicted architectures offer a high localization granularity as well as a low correction latency if extended with Bit-Flipping Latches providing a bitwise correction. The extension to distinguish single from double errors and thereby avoid false corrections further increases the circuit robustness. However, all these advantages are associated with an increased hardware overhead in comparison to sole detection and localization.

Although, the overhead of the presented architectures is already lower than the straight forward application of bitwise dual or triple modular redundancy, hardware and area overhead directly relate to cost while the ability to reduce cost is considered a competitive advantage in economics. This is especially true, as cost is considered crucial and non-functional properties of a design such as safety or reliability, which demand for additional area, are often undervalued. Consequently, the willingness to trade off resources for achieving additional properties raises with the area efficiency.

This chapter focuses on further improving the area efficiency of the fault tolerance architectures. Therefore, the total area overhead from the last chapters is decomposed, attributed to the architectures components and analyzed in the next section. Section 7.2 then optimizes the component identified as the largest overhead contributor - the characteristic computation - by providing and utilizing an area efficient exclusive OR standard cell. The experimental evaluation in Section 7.3 explores the properties of the new standard cell and confirms an improved area efficiency if the cell is applied to multiple components of the fault tolerance architecture.

7.1. Detailed Analysis of the Correction Area Overhead

The area overhead of the Single Error Correction (SEC) architecture (see Sec. 5.3) is now investigated in more detail. [1] Figure 7.1 shows the total area overhead across multiple register sizes, which is decomposed and attributed to the following architecture components: The Bit-Flipping Latches ('BFL'), the characteristic tree ('Char'), the derivation of the syndrome and the syndrome fail signal ('S/fail S'), the protection of the error condition ('fail P'), the syndrome decoder ('Decoder'), as well as the remaining parts such as deriving and latching the correction signal ('Top').

The results for a register size of 63 bits show, that the total area overhead of +228.89 % is partitioned as follows among the different units (Fig. 7.1). The Bit-Flipping Latches entail an overhead of +20 % and the attached characteristic computation adds +108.57 %. Storing the characteristic and deriving the syndrome contributes +15.08 % while its parity protection involves only +6.83 %. Finally, decoding the difference adds +72.22 %. If the overhead is regarded across all register sizes, the following observations can be made. Many architecture components have a negligible overhead. For the components with significant overhead, the characteristic computation and the decoder can be identified as the main contributors. While the overhead for decoding declines with growing register size, the overhead of characteristic computation even increases.

Hence, the characteristic computation offers the highest potential for further area overhead reductions. As mentioned in Section 4.3.2, its overhead depends on two factors: The amount of exclusive OR gates contained in the characteristic tree and the size of each individual gate. By using the optimal tree organization from Section 4.3.2, respectively Section 6.3, the amount of gates within the characteristic tree is already minimal. Thus, a further reduction of the area associated with the characteristic computation is only possible by utilizing a optimized exclusive OR implementation.

7.2. Area Efficient Exclusive OR Trees

The characteristic computation derives the l-bit checksum from a n-bit register by a linear network. It is implemented using XOR2 standard cells from a cell library, which tend to be larger than other combinational cells. For example, in the Open Cell

[1] The analysis is likewise valid for the Single Error Detection (SED) architecture and both architectures targeting double errors (DED and SECDED), which will as well be evaluated in Section 7.3.3.

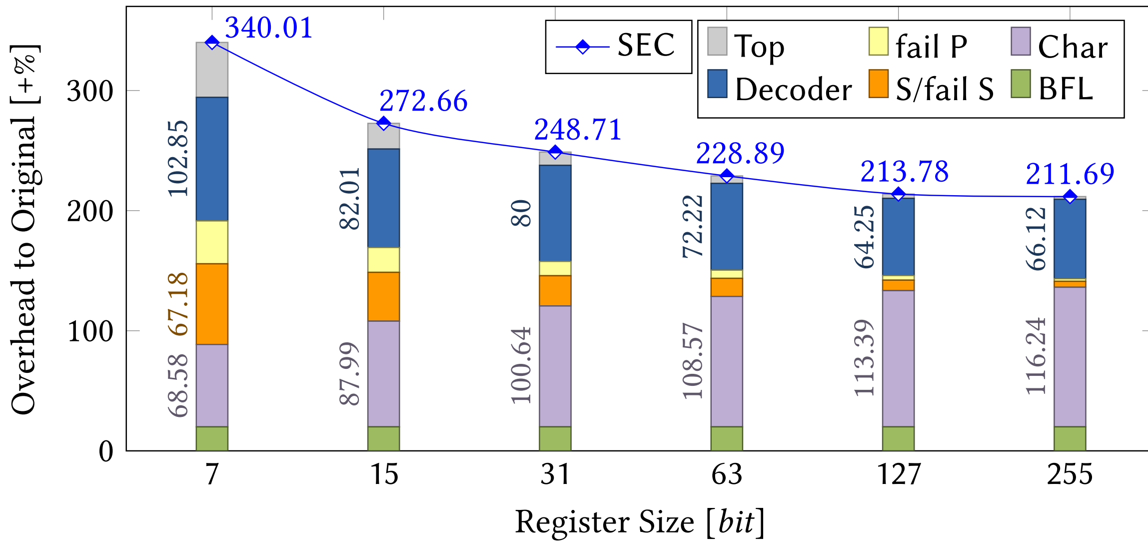

Figure 7.1.: Detailed Area Overhead Analysis of the Single Error Correction (SEC) Components (see Figure 5.9) (data from Table A.9).

Library, the XOR2_X1 standard cell has an area of 1.596 μm^2, while the AND2_X1 cell with 1.064 μm^2 and the NAND2_X1 cell with 0.789 μm^2 are considerably smaller.

The exclusive OR logic function computes the Boolean difference of its inputs as depicted for two inputs by the truth table in Figure 7.2-a. The function can be perceived as a selection between either the identity or the inversion of the first input, which is controlled by the value of the second input.

If the inputs are *dual rail encoded* as shown in Figure 7.2-b both polarities of the first input are available and a cell internal inversion is not necessary. The register

A	B	Z
0	0	0
1	0	1
0	1	1
1	1	0

(a) Single Rail Encoding

A/AN	B/BN	Z
0/1	0/1	0
1/0	0/1	1
0/1	1/0	1
1/0	1/0	0

(b) Input Dual Rail Encoding of the Parity Pair Latch

A/AN	B/BN	Z/ZN
0/1	0/1	0/1
1/0	0/1	1/0
0/1	1/0	1/0
1/0	1/0	0/1

(c) Full Dual Rail Encoding

Figure 7.2.: Exclusive OR Truth Tables.

parity computation from Chapter 4 is one such example, where both polarities are available anyway within the latches. For the first level of the parity tree, the parity computation is efficiently implemented by merging with the latches into *Parity Pair Latches* that require only two additional multiplexers. For the remaining parity tree levels, standard exclusive OR cells have to be used due to the lack of the negated polarities for the intermediate parities.

Hence, in order to facilitate the use of a multiplexer based exclusive OR implementation in exclusive OR trees, it is necessary to implement a full dual rail encoding as depicted in the truth table in Figure 7.2-c which provides the inverted output polarity in addition. This two-input exclusive OR with dual rail encoding for all ports is now implemented by the use of two multiplexers. The existence of an area efficient multiplexer realization by using transmission gates was already depicted within the Parity Pair Latch (Section 4.2.2) as well as the Bit-Flipping Latch (Section 5.3.1).

Figure 7.3 shows the schematic of the presented *Transmission-Gate Exclusive OR*. It uses four transmission gates (labeled TG1 to TG4) to implement the two multiplexers (TG1 and TG2, respectively TG3 and TG4). The input signal pair A/AN is used to drive the output signal pair Z/ZN, whereas the input signal pair B/BN selects the polarity of the connection between A/AN and Z/ZN.

The new exclusive OR can be used to implement exclusive OR trees of arbitrary depth such as the characteristic or parity computation. The expected increase in area efficiency depends on the yet to be proven small footprint of the Transmission-Gate Exclusive OR standard cell implementation.

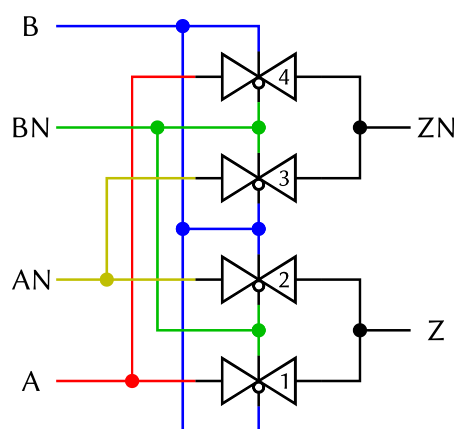

Figure 7.3.: Schematic of the Transmission-Gate Exclusive OR.

7.3. Experimental Evaluation

The Area Efficient Characteristic Computation is now evaluated. The experimental setup is followed by the results for the Transmission-Gate Exclusive OR in Section 7.3.2. Its application to the previously discussed fault tolerance architectures is examined in Section 7.3.3. The employed tools and tabulated results are depicted in Section A.1, respectively Section A.3.4 of the Appendix.

7.3.1. Experimental Setup

The Transmission-Gate Exclusive OR from Section 7.2 is implemented as a new standard cell. Similar to the previous custom standard cells, it is designed according to the design rules and electrical rules of the FreePDK Process Design Kit [SCW+07] to enable compatibility with the Open Cell Library (OCL) [Nan11] cells. Consistency with the schematic is checked and confirmed during *Layout-vs-Schematic (LVS)* and a transistor netlist containing all parasitic layout effects is obtained during *Physical Extraction (PEX)*. The netlists of the Transmission-Gate and the standard Exclusive OR from the Open Cell Library are then simulated at the analog level using SPICE [NP73]. Timing behavior and power consumption are characterized for a rising and a falling edge with a slew rate of 22 V/ns. An inverter (INV_X1) loads all outputs and 10 % respectively 90 % of the nominal voltage are used as trip points.

After *Library Characterization* the standard cell is added to a new library and used to replace most exclusive OR cells in the components of the four previously discussed fault tolerance architectures. In particular, these include the cells in the targeted characteristic computation, its parity protection as well as the syndrome comparator and if present, the register parity protection of the extended characteristic.

7.3.2. Transmission-Gate Exclusive OR Standard Cell

This section discusses the area, the timing behavior as well as the power consumption and energy of the Transmission-Gate Exclusive OR standard cell.

Standard Cell Area

The Transmission-Gate Exclusive OR uses 8 transistors which are placed as depicted on the left hand side of Figure 7.4. Thereby, the cell width is reduced as only two straight polysilicon lines are required. The final layout including the cell internal metal wiring is shown on the right hand side. The TGXOR_X1 cell has a width of 0.57 μm and an OCL compatible height of 1.4 μm.

Compared to the OCL XOR_X1 cell with an area of 1.596 μm^2, the TGXOR_X1 cell has an area of 0.798 μm^2 and thereby enables a significant reduction of 50 %. Hence, the architecture's area efficiency can be improved, especially when taking into account the high fraction of employed exclusive OR cells.

Timing Behavior and Signal Quality

In the desired application to fault tolerance architectures, the exclusive OR is used in the well defined environment of parity trees. Thus, the cell load is bound as only XORs can occur as a load and the fanout at the cell output is limited. Therefore, a tree fragment of a series of three XORs is simulated, where input A is on the sensitized path and input B has a constant value of 0.

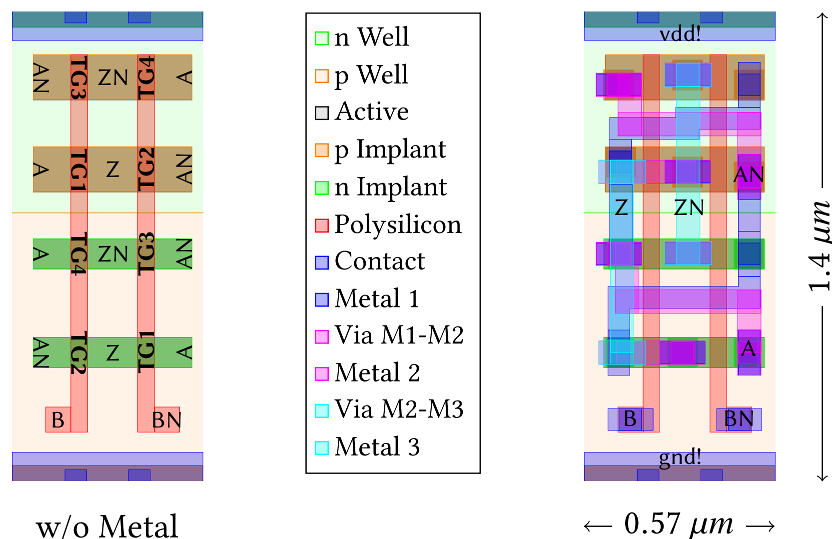

Figure 7.4.: Layout of the Transmission-Gate Exclusive OR Standard Cell TGXOR_X1.

The Timing Behavior is depicted in Figure 7.5 for TGXOR_X1 as well as OCL XOR_X1 cells. For the OCL implementation, the first cell has a delay of 112.01 *ps* for a rising and 83.05 *ps* for a falling transition. After three cells, the rising transition is delayed by 208.4 *ps* and the falling transition by 161.33 *ps*. The TGXOR_X1 implementation with 57.21 *ps* (rising) and 44.24 *ps* (falling) is much faster after the first level. Actually, with 71.87 *ps* respectively 63.05 *ps* after the third level, the delay is even lower than for a single level of OCL XORs.

This dramatic speed increase is explained by the intended non-compliance to the CMOS property of actively driving the cell output by either the supply or the ground voltage. While this could generally result in a degraded signal quality for highly loaded outputs, the confined field of application circumvents this corner case and a degradation of signal quality could not be observed in the performed simulations.

Power Consumption and Energy

The power consumption of the reference and the Transmission-Gate Exclusive OR are depicted in Figure 7.6. The faster switching speed of the Transmission-Gate Exclusive OR demands a raised *peak power consumption* of 103.7 μW for the TGXOR_X1 cell in contrast to the 58.1 μW for the reference XOR_X1 cell. Diametrical, the increased

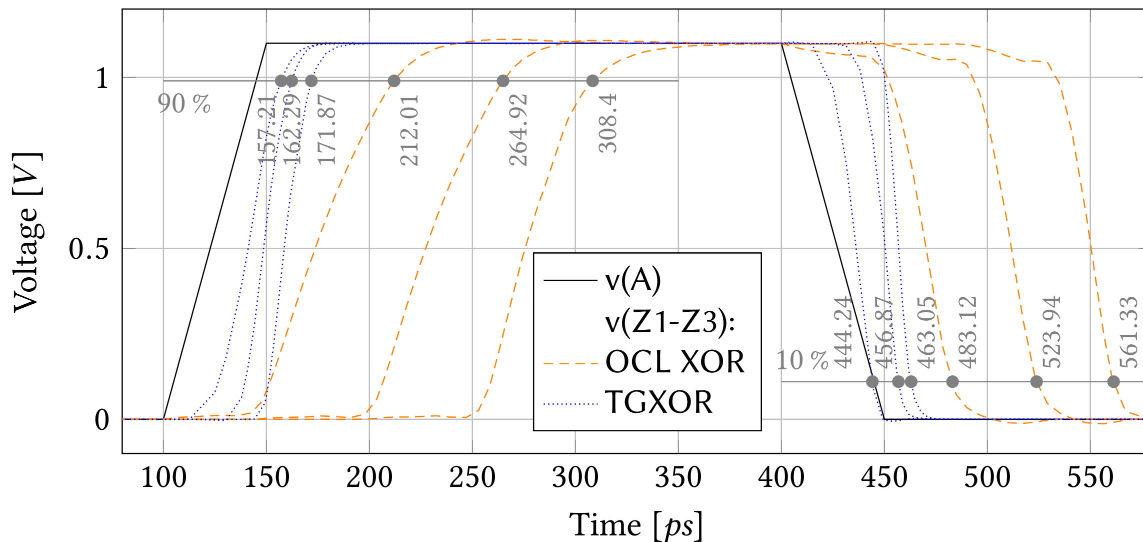

Figure 7.5.: Timing Behavior of the OCL Exclusive OR (XOR2_X1) and the Transmission-Gate Exclusive OR (TGXOR2_X1).

speed is beneficial for the *average power consumption* as power is consumed in a much shorter time period that compensates for the higher instantaneous power. With an average power of 19.20 μW for the XOR2_X1 and 9.95 μW for the TGXOR_X1 cell, the average power consumption is almost bisected to 51.8 %. The *energy* during the simulated 600 ps reflects the same reduction and is lowered from 115 pJ to 59.7 pJ.

7.3.3. Area Efficient Characteristic Computation

The developed Transmission-Gate Exclusive OR cell is now used to improve the area efficiency of the single error correction fault tolerance scheme. Therefore, the cell is as intended deployed to the characteristic tree and used in addition to optimize the syndrome comparator as well as the protected storage of the error condition.

The resulting Area Efficient Characteristic Computation is now first evaluated for single registers equipped with the Single Error Correction (SEC) which was analyzed at the beginning of the chapter (see Section 7.1). The results depicted in Figure 7.7 show, that the area overhead of the characteristic computation is as expected reduced by 50 %. Additional savings arise from the application to the syndrome comparator ('fail S') as well as the optimized protection of the error condition ('fail P'). For a

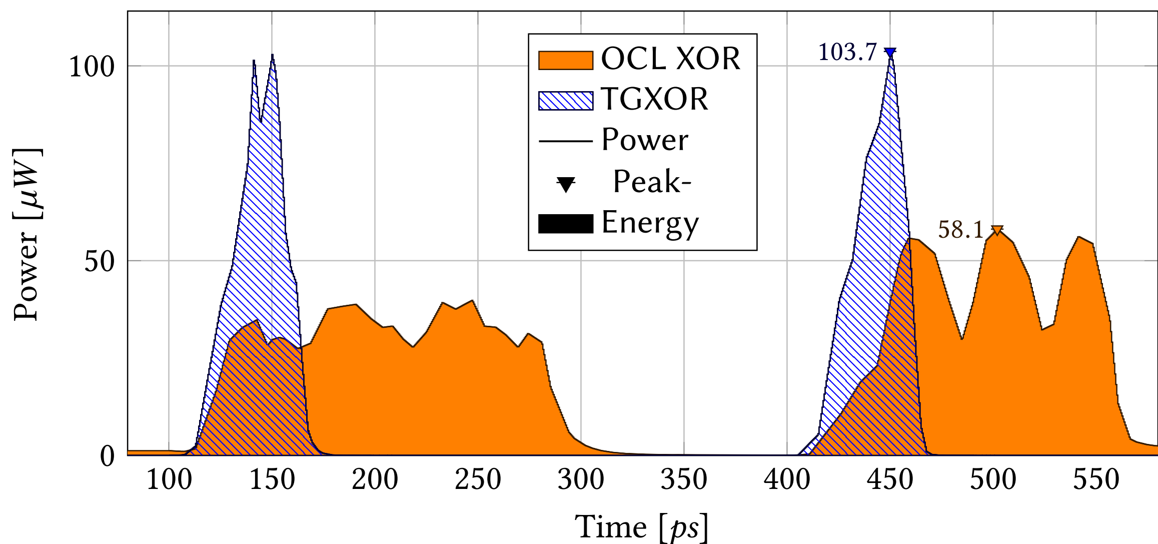

Figure 7.6.: Power and Energy of the OCL Exclusive OR (XOR2_X1) and the Transmission-Gate Exclusive OR (TGXOR2_X1).

63-bit register, the area overhead of +228.89 % is reduced to +170.16 %. Thus, fault tolerance is achieved with even less overhead than sole detection implemented by bitwise application of Duplication with Comparison (DWC).

The Transmission-Gate Exclusive OR is also applicable to the Single Error Detection (SED) from Section 5.2 as well as the extended characteristic $C+$ employed in the Double Error Detection (DED) and the Single Error Correction Double Error Detection (SECDED) architectures from Chapter 6. The results for different register sizes depicted in Figure 7.8 disclose an area overhead reduction of more than 50 % independent of the register size or used architecture configuration.

7.4. Summary

The analysis of the area overhead associated with the Single Error Correction scheme from Chapter 5 identified the characteristic computation as a major area contributor. Due to the amount of used Exclusive OR gates being already minimal, the standard cell implementation itself constitutes the last resort for further area reduction.

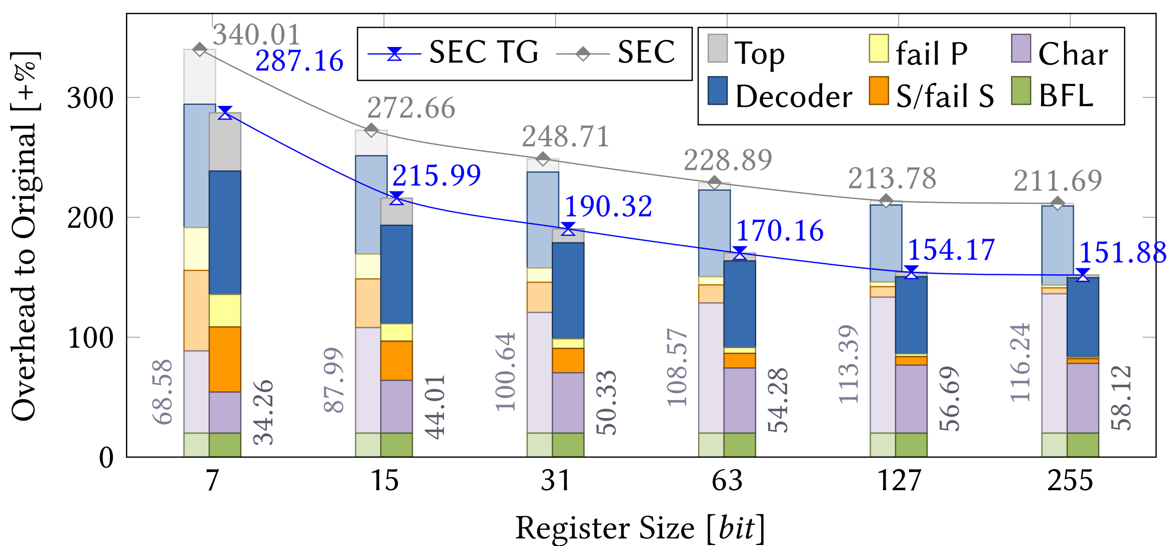

Figure 7.7.: Detailed Area Overhead Analysis of the Area Efficient Single Error Correction (SEC TG) utilizing the Transmission-Gate XOR (data from Table A.10).

The use of a dual rail encoding within the exclusive OR trees enables a standard cell implementation that only employs multiplexers. The presented *Transmission-Gate Exclusive OR* depicts an area efficient realization containing four transmission gates. The experimental evaluation quantifies a significant cell area reduction by 50 %. As a side effect, the cell is substantially faster by more than 3 X and almost bisects the average power consumption and energy.

The cell's application to multiple building blocks of the previously discussed fault tolerance architectures reconfirms the targeted area efficiency increase and lowers the area overhead by at least 50 %. For registers with 63 or more bits, the bitwise implementation of Dual Modular Redundancy is even more expensive than Error Correction based on the logarithmic characteristic.

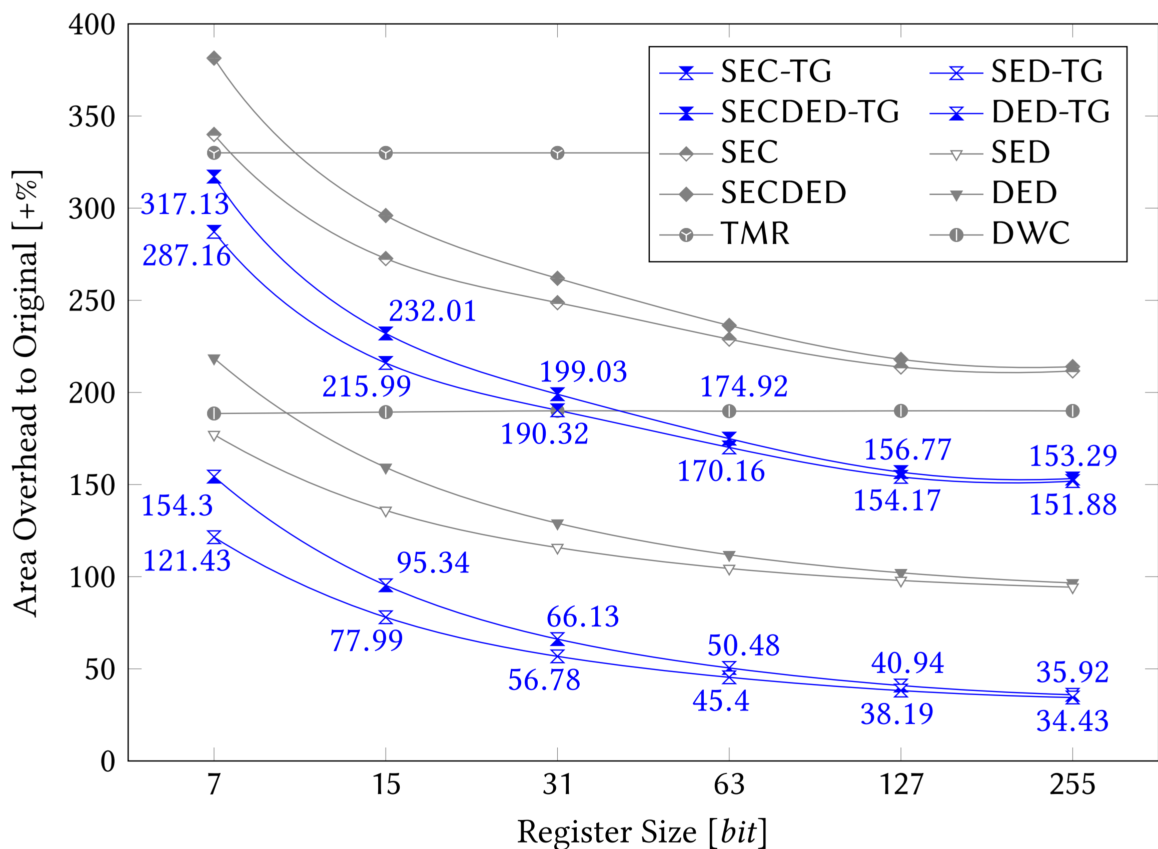

Figure 7.8.: Area Overhead - Area Efficient Architectures (SED TG, DED TG, SEC TG, SECDED TG) - Single Register (data from Table A.11 and Table A.12).

Summary and Discussion of Part II

This part targeted soft errors in the sequential state of a circuit's random logic by detection, localization and correction implemented with different localization granularities and correction capabilities. In favor of a bitwise redundancy implementation, error detecting and correcting codes are used across all presented fault tolerance architectures which will be recapitulated in the following.

Chapter 4 - Non-Concurrent Detection and Localization of Single Event Upsets - focuses on circuits equipped with clock gating to diminish the power consumption during idle phases. During these clock gated phases, the sequential state of a module is retained over long periods of time. Thus, assuring the data correctness upon leaving the gated phase is a necessity. The presented two-tiered architecture combines an efficient error detection based on individual register parities with a module-wide localization founded on a logarithmic checksum of register parities. The *Parity Pair Latch* efficiently implements the detection by merging the register latches with the first level of the parity tree into a new standard cell. Compared to the reference implementation, it bisects the area overhead and significantly accelerates the parity computation while reducing the power consumption and energy considerably. The *Modulo-2 Address Characteristic* of register parities is used for module-wide localization and inherits a low gate and connection count by using the optimal characteristic tree organization. Overall, for modules with registers containing 16 or more bits, the area overhead associated with detection and localization is reduced from over +90 % to below +20 %.

Chapter 5 - Concurrent Online Correction of Single Event Upsets - targets single errors affecting the sequential state during the operational phase of a circuit. *Single Error Detection* is achieved by implementing the *modulo-2 address characteristic* for individual registers to derive a register specific error condition. The protected storage of the error condition eliminates false detections that may result from soft errors directly affecting the architecture while the register content is correct. Thus, all single errors are detected and localized and can be corrected by recomputation. Compared

to bitwise Dual Modular Redundancy, the use of a logarithmic error detecting and correcting code results in a lowered area overhead which is almost bisected for larger registers. *Single Error Correction* within one clock cycle is achieved by exploiting the computed localization information. It is used to control the efficient low level correction provided by the *Bit-Flipping Latch* standard cell which augments a latch with the ability to invert its stored value. The Bit-Flipping Latch requires only +20 % area in addition to a latch, has no negative impact on timing behavior of the data path and reduces the average power consumption and energy by 25 %. The self-contained online correction architecture has a time vulnerability factor, i.e. the unprotected time interval of a register, of zero. Compared to bitwise Triple Modular Redundancy, the area overhead is reduced by more than one third for larger registers.

Chapter 6 - Fault Tolerance in Presence of Multiple Bit Upsets - contemplates the behavior of the online architecture under *Multiple Bit Upsets*. While double errors affecting the register are correctly detected, their localization and thus correction is not possible due to the limited minimum Hamming distance of the used characteristic. However, the introduction of an additional register parity bit allows to distinguish correctable single errors from double errors and completely avoids false corrections. The resulting *Extended Characteristic C+* is computed with negligible hardware overhead by reusing intermediate results of the optimal characteristic tree implementation. Thus, *Single and Double Error Detection* as well as *Single Error Correction Double Error Detection* are facilitated at a lower area overhead than bitwise modular redundancy for register with 15 or more bits while retaining all other architecture properties.

Chapter 7 - Area Efficient Characteristic Computation - analyzes the architecture's area overhead in more detail and identifies the characteristic computation as a major area overhead contributor. Due to the already optimal characteristic tree organization, a further area reduction can only be expected through an optimized Exclusive OR standard cell. The presented *Transmission-Gate Exclusive OR* is smaller by 50 %, faster by three times and has a lower power consumption and energy than the reference Exclusive OR. Its application to the characteristic tree as well as other building blocks of the presented fault tolerance architectures lowers their previously reported area overheads by at least 50 % in all cases. Thereby, for register with 64 or more bits, fault tolerance based on the presented error correction utilizing a logarithmic checksum becomes even more favorable in terms of hardware overhead than sole detection achieved through the bitwise implementation of Double Modular Redundancy.

Part III

Infrastructure Reuse for Offline Testing

Chapter 8
Test Access through Infrastructure Reuse

The emerging need for fault tolerance was targeted in the last part by the introduction of a self-contained infrastructure able to correct Single Bit Upsets in the sequential circuit state. Typically, a design is augmented by different types of infrastructure serving orthogonal objectives. The most widely used field that utilizes on-chip infrastructure is *test*, an experiment to show the presence of hard faults, which is performed at least once for every produced chip during *manufacturing test*. Testing a circuit involves the abstraction from defects to faults within a fault model and the generation of a circuit specific test set that covers a maximized fraction of all possible fault locations. As faults located at internal nodes may exhibit a low accessibility, usually additional infrastructure is employed that serves as a *Test Access Mechanism (TAM)* in order to increase testability and reduce test application time at the cost of additional area. The traditional use of two distinct infrastructures to conquer soft errors during operation while orthogonally providing test access raises the chip area that has to be allotted to infrastructure. Collaterally, both infrastructures are never used concurrently due to the contradicting goals pursued by fault tolerance, that mitigates the effect of soft errors, and test, that shows the presence of hard faults.

This chapter is based on the Bit-Flipping Scan architecture from [IW14], a unified infrastructure with low hardware overhead that procures fault tolerance by online correction and supports offline test by serving as an efficient test access mechanism. The remainder of this chapter starts by depicting the unified architecture and its extensions in excess of the online fault tolerance from the previous part. Subsequently, test application under infrastructure reuse is detailed along with the modes of operation that provide test access to the sequential circuit state in Sections 8.2 to 8.4. Finally, the reachable test access efficiency is discussed theoretically before concluding with a short summary.

8.1. Unified Architecture

Traditionally, test access is provided by the introduction of scan design where the circuit registers are replaced by scan chains. The presented unified architecture implements test access in addition to fault tolerance by exploiting similarities between both fields. Fault tolerance involves the introduction of redundancy to detect and localize undesired register content modifications in combination with an efficient correction mechanism during circuit operation. Test requires to observe the circuit response to a test for comparison with the expected test response in addition to control of the circuit state to set up and apply the next test.

The unified architecture depicted in Figure 8.1 is based on the online fault tolerance architecture from Chapter 5. Instead of providing direct test access to each register R_i with the help of scan design, test access to register R_i is indirectly enabled via its associated characteristic register C_i. The two fundamental concepts of the previously discussed fault tolerance architecture, namely the *logarithmic register checksum* and the *bit-flipping mechanism*, are reused to provide full access to register R_i and facilitate observability and controllability. The external test interface is implemented by outfitting the characteristic register C_i with scan design in order to supervise both concepts during test mode.

Only small modifications in excess to the already discussed self contained fault tolerance architecture are necessary to demarcate the application of the unified architecture as a test access mechanism which will be discussed along with its two essential operation modes in the next three chapters.

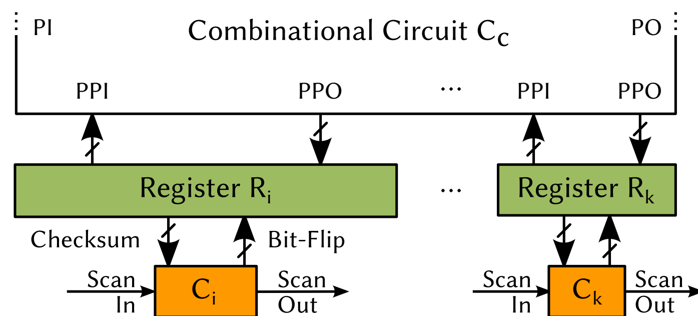

Figure 8.1.: Unified Architecture.

8.2. Test Application

A brief overview on the sequence of operations used during test application will be given to clarify the similarities and differences between classical scan design and Bit-Flipping Scan with a special focus on the used test access mechanisms. Suppose a sequential circuit with at least two registers R_i and R_k as depicted in Figure 8.1. Let p_1 and p_2 denote two consecutive patterns of a test sequence where p_2 targets a fault f. Further let f be controllable through register R_i (at least one bit of R_i is contained in the input cone or in the support of f, see Figure 2.1) and observable via register R_k (the output cone of f feeds at least one bit of R_k).

Classical Scan Design equips all registers with scan chains (see Section 2.3.2). In order to set up pattern p_2, the next state of register R_i is serially shifted into the corresponding scan chain. One capture cycle is applied to record the sequential portion of the test response into the registers. To assert the presence of fault f, the content of register R_k is shifted out and compared to the reference response.

Bit-Flipping Scan assumes a protection by the unified architecture. Thus, all registers contain the checksum computation, the bit-flipping mechanism, and an additional characteristic register. To apply a test pattern, a sequence of operations is used as follows (Figure 8.2). After pattern p_1, the checksum contained in C_i is used to validate the state of register R_i (③). To set up test pattern p_2, an incremental update of the known and unaltered state of R_i is performed by a series of bit-flips (③-⑥). For each bit position to be updated, the checksum of the next desired register state is shifted into C_i (③,⑤) to trigger the inversion of that register bit (④,⑥). After R_i is updated (⑥), the test response is captured to R_k (⑦) and validated via C_k (⑧,⑨).

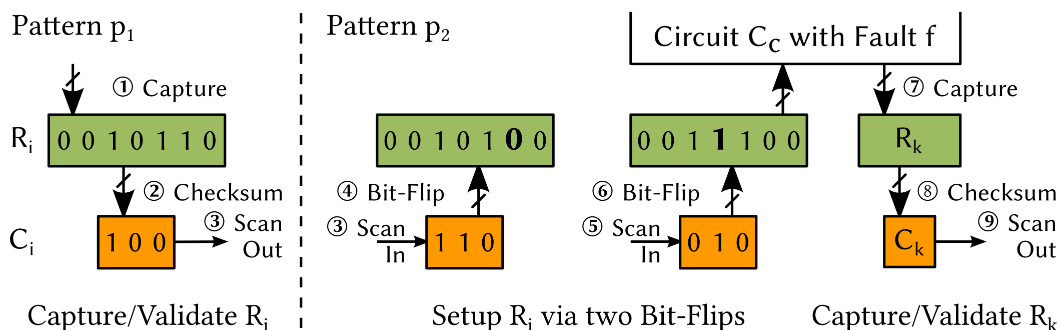

Figure 8.2.: Bit-Flipping Scan Test Application.

8.3. Observing a Test Response

After setting up all test conditions for a test pattern p, the circuit response is captured into the internal registers. In order to validate if the test pattern passed or failed, the test response must be observed. The fault tolerance infrastructure already computes the reference characteristic C and stores it in the additional register C_i (see Section 5.2.1). Figure 8.3 shows the architecture parts reused for observing the register content.

The value of C_i depends on all bits of register R_i and represents a compacted version of the register content. Instead of directly observing R_i by adding it to a scan chain and shifting out all contained bits in n shift cycles, C_i is made scannable and the compacted circuit response can be observed in l shift cycles, where $l \ll n$ (see Eq. (4.6)).

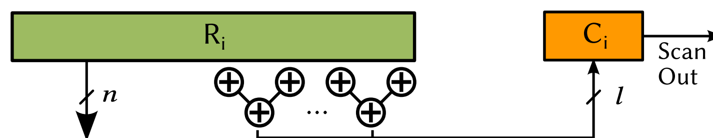

Figure 8.3.: Observing the Compacted Test Response of a Register.

Extending the reference characteristic register C_i with scan design would lead to false corrections in register R_i as any shift operation of C_i will violate the entanglement of the register R_i and its reference checksum C_i. To resolve this situation the occurrence of not intended bit flips is avoided

- during shift operations by gating the syndrome decoder with the scan enable signal and
- after completion of the scan operation, when the decoder is no longer gated, by shifting in the expected reference characteristic during the scan operation.

Note, that the presented architecture implicitly shifts out the compacted test response after each pattern and every bit-flip. Hence, the complete sequential state is always observed with low overhead as opposed to other architectures, where every observed bit increases the test time (Random Access Scan, see Sec. 3.2.2) or the parallel readout demands for additional area (Progressive Random Access Scan, see Sec. 3.2.2).

8.4. Controlling a Register by Bit-Flipping

A mechanism to flip single bits of a register R_i is present in the architecture to correct SBUs in the fault tolerance mode (see Section 5.3.1). Now, this feature is used to set up the next state of register R_i by a series of bit-flips. Figure 8.4 shows the involved architecture parts.

Let p_1 and p_2 denote two consecutive test patterns of a test sequence. Further, let $O(R_i,p_1)$ denote the state of register R_i after the capture cycle of p_1 with both characteristics C_i, C_i' being equal and let $I(R_i,p_2)$ denote the state of R_i needed to set up p_2. Assume without loss of generality, that $I(R_i,p_2)$ and $O(R_i,p_1)$ differ in exactly one bit at address $adr\ b$ ($1 \leq adr\ b \leq n$), and their Hamming distance is one. Then, the desired register value can be deduced by a single flip of the bit at $adr\ b$.

In order to perform a bit-flip the syndrome computed on-chip needs to result in the address of the bit to be flipped.

$$S_i := adr\ b$$

As the register state after p_1 and thus the associated recomputed characteristic C_i' are known, the reference characteristic C_i can be predetermined.

$$C_i := S_i \oplus C_i'$$

Scanning in C_i triggers a bit-flip at address $adr\ b$ and thus generates $I(R_i,p_2)$ from $O(R_i,p_1)$ with l shift cycles and one additional cycle for the bit-flipping. At the same time, the compacted register state C_i is shifted out and observed.

If the Hamming distance between the two register states is larger than one, a series of single bit-flips is used.

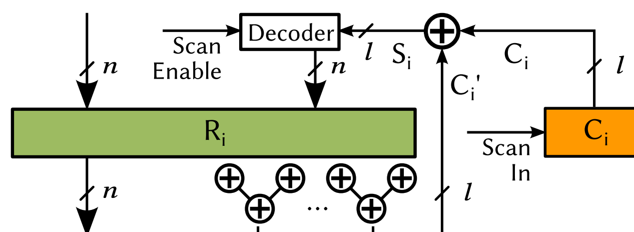

Figure 8.4.: Controlling a Register by Bit-Flipping.

8.5. Test Access Efficiency

The *test application time (TAT)* for scan design depends on the maximum scan chain length n and the pattern count. To apply a single pattern p_2 the captured response $O(R_i, p_1)$ of the previous pattern p_1 is shifted out in n cycles while the desired state $I(R_i, p_2)$ for p_2 is shifted in. Then the test response is captured in one extra cycle.

$$TAT_S = n + 1$$

For the presented bit-flipping scan scheme, the test time is dominated by the number of bit-flips bf. For each flip, l shift cycles and one flip cycle are needed. After applying all flips the circuit state is captured.

$$TAT_{BFS} = bf \cdot (l + 1) + 1$$

In order to reduce the test application time compared to scan design, the maximum number of flips per pattern is bound by

$$\begin{aligned} TAT_{BFS} &\leq TAT_S \\ \Leftrightarrow bf \cdot (l + 1) + 1 &\leq n + 1 \\ \Leftrightarrow bf &\leq \frac{n}{l + 1} \\ \Leftrightarrow bf &\leq \frac{n}{\lceil log_2(n + 1) \rceil + 1} \,. \end{aligned} \tag{8.1}$$

Thus, for a maximum register size of 127 bits, bit-flipping scan still results in a lower TAT than standard scan design with a maximum chain length of 127 bits if no more than 15 flips are required per register and pattern ($bf \leq 15.875$).

8.6. Summary

Test since ever is a necessity to show the presence of hard faults. To ease test execution testability is increased by augmenting circuits with test access infrastructure. The presented *Unified Architecture* provides test access by reusing the two fundamental concepts of the previously discussed fault tolerance architecture. The *modulo-2 address characteristic* enables full observability of the register content with logarithmic effort while *bit-flipping* allows an incremental register update of crucial bit positions in the captured circuit answer and thereby offers full controllability.

Chapter 9
Test Sequence Generation

The Bit-Flipping Scan architecture provides an efficient test access mechanism that enables the use of arbitrary test sets generated by ATPG. During test conduction, the sequential circuit state is incrementally updated through a series of bit-flips that transform the present state after the last test pattern into the initial state of the following test pattern. Thus, the time to apply the complete test set is dominated by the amount of necessary updates which is determined by the sequential state specified within the test patterns. As a test set that targets scan-design is generated under the implication of full accessibility to the sequential state by combinational ATPG, its fully specified patterns are unordered. Hence, the *Test Application Time* can be reduced by test set reordering that minimizes the amount of necessary bit-flips.

However, test time is still bound by the sequential states specified during test generation. To this end, the amount of specified bits in the test set can be reduced either during ATPG or through test set stripping in a post-processing step [KZIW08]. Consequently, appropriate filling of unspecified bits allows to avoid irrelevant bit-flips.

Ideally, to put the architecture's bit-flipping capability into practice and demonstrate its full benefits, a tailored test sequence is needed that

- ▹ reuses the sequential state of the previous pattern,
- ▹ limits the amount of necessary bit-flips per pattern while it
- ▹ maximizes the number of faults covered per pattern and
- ▹ avoids unnecessary bit-flips whenever they do not contribute to fault coverage.

This chapter builds upon the sequential test sequence generation from [IW14] that targets the unified architecture by explicitly modeling its bit-flipping capability and is able to produce tailored test sequences fulfilling the above mentioned properties.

The algorithm iteratively generates the test sequence as depicted in Figure 9.1. Therefore, the circuit structure, the faults to be covered, and the bit-flipping ability within the sequential state are modeled. The model is augmented with cardinality constraints to control the number of bit-flips as well as the amount of faults covered. Starting from the current sequential state a pattern is generated with precedence on improving the fault coverage while using a minimal amount of bit-flips. Consecutively, the coverage of all modeled faults is optimized under the laid bit-flip bound. Finally, the pattern is accepted and the procedure is repeated until all faults are covered.

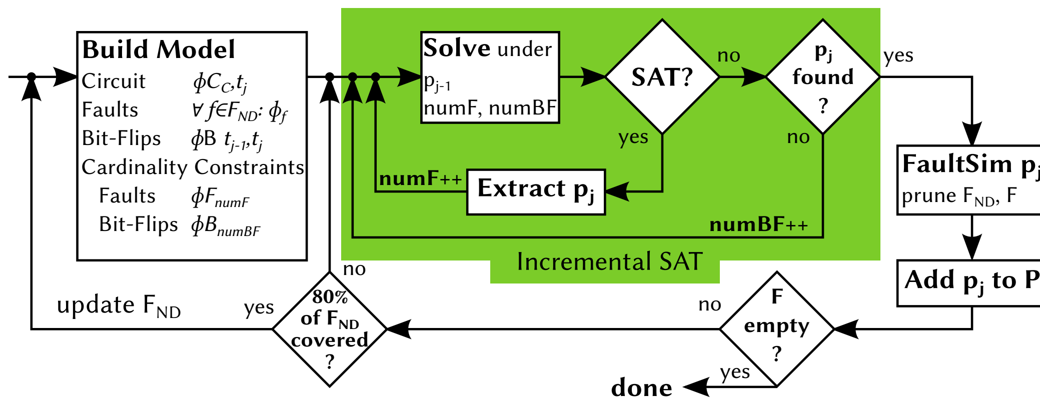

Figure 9.1.: Iterative Bit-Flipping Scan Test Pattern Generation.

The remainder of this chapter starts with details on the used models in Section 9.1 before discussing the challenges in finding an optimal test sequence in Section 9.2. The iterative Bit-Flipping Scan test sequence generation is detailed in Section 9.3 and evaluated with respect to test time, volume, power and energy in Chapter 10.

9.1. Modeling the Test Sequence Generation

For a sequential circuit C_S with a set of faults F, an optimal bit-flipping scan test sequence P_{opt} ensures that

- all faults f in the fault universe F are detected by P_{opt},
- the number of bit-flips to set up a register R_i for pattern p_i from the previous registers state $O(R_i, t_{j-1})$ is bound by $\Delta_H(O(R_i, t_{j-1}), I(R_i, t_j)) \leq bf_{bound}$, and
- the length of P_{opt} is minimal.

Finding such a test sequence is modeled as a Boolean satisfiability problem in *conjunctive normal form (CNF)* (see Section 2.4).

9.1.1. Circuit Modeling

The combinational core C_C of the sequential circuit C_S is extracted by removing all sequential elements and adding pseudo-primary in- and outputs (PPI/PPO). The circuit model in conjunctive normal form is then obtained by applying the *Tseitin transformation* which yields a model with a linear number of clauses at the cost of introducing a linear number of new literals [Tse68; Tse83].

To derive the CNF model Φ_{C_C} for each circuit input $i \in I$, output $o \in O$i, and internal signal $s \in S$, a new literal is added to the set of literals: $L = \{I, O, S\}$.

Each gate $g \in G_{C_C}$ with inputs $(i_1, \cdots, i_n)$ and output o implementing a Boolean function $o = \phi\,(i_1, \cdots, i_n)$ is logically equivalent to

$$\begin{aligned} g &\equiv (o \rightarrow \phi\,(i_1, \cdots, i_n)) \wedge (\phi\,(i_1, \cdots, i_n) \rightarrow o) \\ &\equiv (\overline{o} \vee \phi\,(i_1, \cdots, i_n)) \wedge \left(\overline{\phi\,(i_1, \cdots, i_n)} \vee o\right) \quad . \end{aligned}$$

Expanding the equation in a product-of-sums form yields the set of clauses Φ_g in CNF. The circuit C_C is then described in CNF as

$$\Phi_{C_C} = \bigwedge_{g \in GC_C} \Phi_g \quad . \tag{9.1}$$

9.1.2. Fault Modeling

To represent stuck-at faults, the circuit model is structurally extended in order to distinguish the circuit behavior in absence of the fault (good circuit) from the behavior in presence of the fault (faulty circuit) as shown in Figure 9.2.

Given a stuck-at fault f located at signal $s_f \in \{I, S, O\}$. First, the model of the fault cone Φ_{c_f} (output cone of the fault site) is copied to derive the faulty circuit model $\Phi_{c'_f}$ under f. New literals are assigned for the fault location and all other signals in the cloned cone (s'_f for s_f and $\forall s_n \in c_f : s'_n$). All signals at the edge of the cone are connected to the signals in the good circuit.

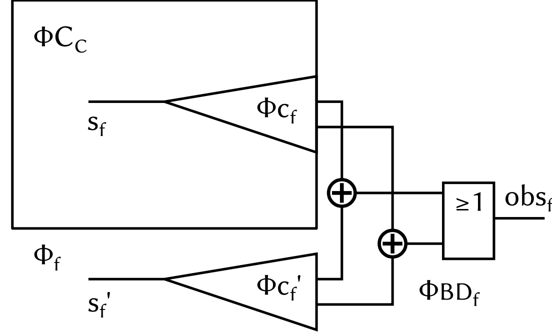

Figure 9.2.: Model of a Combinational Circuit ΦC_C (Single Timeframe) and a Target Fault f represented by Φ_f .

Second, the Boolean difference Φ_{BD_f} between the faulty and the fault free circuit is modeled as the bitwise comparison of all outputs in the faulty cone c'_f with the corresponding outputs in the fault free cone c_f.

$$\Phi_{BD_f} = obs_f \Leftrightarrow \bigvee_{\forall (o,o') \in (c_f, c'_f)} (o \oplus o') \tag{9.2}$$

In addition to the structural representation of fault f with polarity $p_f \in \{0,1\}$, three constraints need to be fulfilled to generate a test pattern:

- ▷ In the fault-free circuit, the fault location carries the correct value: $s_f := \overline{p_f}$.
- ▷ In the faulty circuit, the fault location carries the faulty value: $s'_f := p_f$.
- ▷ The fault effect is observed in at least one output: $obs_f = 1$.

Then fault f is modeled as

$$\begin{aligned} \Phi_f = \Phi_{c'_f} \wedge \Phi_{BD_f} \wedge \Big(f \vee (\overline{s_f = \overline{p_f}}) \vee (\overline{s'_f = p_f}) \vee (\overline{obs_f})\Big) \wedge \\ (\overline{f} \vee (s_f = \overline{p_f})) \wedge (\overline{f} \vee (s'_f = p_f)) \wedge (\overline{f} \vee obs_f) \quad . \end{aligned} \tag{9.3}$$

9.1.3. Sequential Mapping and Modeling of Bit-Flips

The sequential behavior of C_S is modeled by unrolling as shown in Figure 9.3. Each timeframe t_j is modeled by Φ_{C_C,t_j} consisting of a copy of Φ_{C_C} with appropriate literal renaming and Φ_{f,t_j} denotes fault f in timeframe t_j.

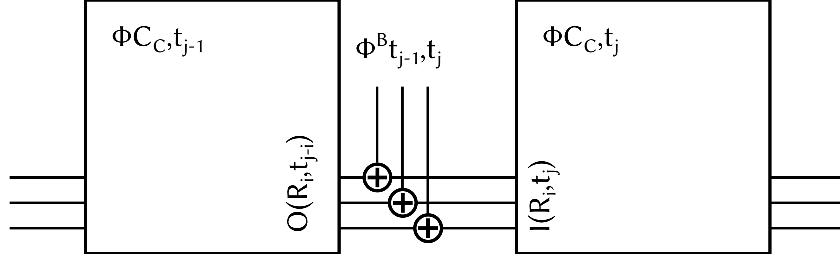

Figure 9.3.: Sequential Mapping Modeled by Unrolled Timeframes $\Phi C_C, t_{j-1}, \Phi C_C, t_j$ and Model of Bit-Flips $\Phi^B_{t_{j-1},t_j}$.

Bit-Flips are modeled by introducing new free literals $B(R_i, t_j)$ for each register R_i. Together with the pseudo-primary output literals $O(R_i, t_{j-1})$ of the previous timeframe the sequential state in timeframe t_j is modeled.

$$\Phi^B_{t_{j-1},t_j} = \bigwedge_{\forall R_i \in C_S} \Big(O(R_i, t_{j-1}) \oplus B(R_i, t_j) = I(R_i, t_j) \Big) \tag{9.4}$$

For an unrolled model with x timeframes, the sequential behavior under bit-flips is modeled by Φ_B.

$$\Phi_B = \bigwedge_{\forall j=1\ldots x} \Phi^B_{t_{j-1},t_j} \tag{9.5}$$

The number of bit-flips is restricted by a cardinality constraint that allows 'atmost' bf_{bound} flip literals per register and timeframe to be true.

$$\Phi^{B_{card}}_{t_{j-1},t_j} = \bigwedge_{\forall R_i \in C_S} atmost(B(R_i, t_j), bf_{bound}) \tag{9.6}$$

9.2. Optimal Test Sequence

The SAT instance for the optimal test sequence P_{opt} from the beginning of this section can now be modeled as follows. The circuit is unrolled for x timeframes and the combinational core is modeled per timeframe (Eq. (9.7)). All target faults are modeled in each timeframe. As it is sufficient to detect a fault once, a disjunction over all timeframes is added per fault (Eq. (9.8)).

$$\Phi_C = \bigwedge_{\forall j=1\ldots x} \Phi_{C_C, t_j} \tag{9.7}$$

$$\Phi_F = \left(\bigwedge_{\forall t_j \forall f} \left(\Phi_{f,t_j} \right) \right) \bigwedge \left(\bigwedge_{\forall f} \left(\bigvee_{\forall t_j} \left(f(t_j) \right) \right) \right) \tag{9.8}$$

The sequential behavior under bit-flips is modeled by Φ_B (Eq. (9.5)) and between consecutive timeframes, bit-flip cardinality constraints $\Phi_{B_{card}}$ are added to limit the maximum number of flips per register (Eq. 9.9). The literals of timeframe 0 are set to the registers' initialization values in Φ_0.

$$\Phi_{B_{card}} = \bigwedge_{\forall j=1}^{x} \left(\Phi^{B_{card}}_{t_{j-1},t_j} \right) \tag{9.9}$$

Solving the model $\Phi_{opt} = \Phi_C \wedge \Phi_F \wedge \Phi_B \wedge \Phi_{B_{card}} \wedge \Phi_0$ yields a solution for x timeframes if it exists. The assignment of literals associated with primary in- and outputs in each timeframe t_i corresponds to a pattern p_i. The generated sequence detects all faults in F with at most bf_{bound} bit-flips per pattern. The test sequence P_{opt} with minimum length can be found by bisection over the number of timeframes.

Finding the optimal test sequence is only feasible for small circuits, small fault universes and a limited number of timeframes due to the high complexity of ATPG and the associated runtimes [IS75]. Nonetheless, an optimized bit-flipping scan test sequence can be generated iteratively as depicted in the next section.

9.3. Bit-Flipping Scan Test Sequence Generation

The heuristic iteratively generates patterns of the test sequence where each pattern targets a limited number of faults F_{ND} from the fault universe F. Each pattern p is guaranteed to require a minimal number of bit-flips while covering the maximum amount of faults from F_{ND}.

To apply the heuristic three preprocessing steps are performed. First, the single-fault theorem is exploited which testifies that "a target single-fault [from F] that is untestable in [the combinational core C_C] is also untestable in the sequential circuit [C]" [AC95]. Consequently, all *detectable faults* are identified by combinational fault grading and all other faults are pruned from the fault universe F. Second, *easy faults* with a high probability of being covered by random patterns are targeted by a sequence of test patterns which ensures that all register bits at least once carried both

logic values. For all registers and all contained PPIs the assignment of 0 and 1 values is tracked. The bit values the initial sequential state are marked. Each subsequent pattern starts from the current sequential state and performs one bit-flip per register such that the selected bit has a not yet covered value after the flip. The value is marked as covered and the PIs are randomly assigned. The detected faults are removed from F and the bit values of the resulting sequential state are marked. Additional patterns are generated until all PPI values are covered. The length of the generated sequence is limited by the width of the largest register. At last, the remaining *difficult faults* are sorted in descending order according to their testability to put preference on hard to detect faults, and are targeted by the iterative pattern generation discussed next.

The heuristic depicted in pseudocode in Algorithm 1 is called with the pattern sequence P from the preprocessing, the index j of the last pattern, the fault universe F and the maximum number of concurrently targeted faults *maxF*. The algorithm starts by initializing the list of targeted faults F_{ND}. Second, the SAT-model Φ (l. 4) is built that consists of one timeframe of the combinational circuit (Φ_{C_C}), the limited number of faults contained in F_{ND} as well as the bit-flip literals associated to the PPIs Φ_B. Last, the amount of bit-flips allowed per register *numBF* is modeled as $\Phi_{B_{numBF}}$.

The for-loop (l. 6-14) searches for a pattern p_j that covers a maximum number of faults *numF* from F_{ND} while respecting the fixed amount of bit-flips allows by *numBF*. The cardinality constraint $\Phi_{F_{numF}} = atleast(F_{ND}, numF)$ is used to require the detection of at least *numF* faults. If the model is satisfiable under the current sequential state $\Phi_{p_{j-1}}$ and the two constraints, the found pattern p_j is extracted and the loop continues with a constraint tightened by incrementing *numF*. If the model is not satisfiable, two cases need to be distinguished.

In the first case, a pattern was found (l. 15) and the current iteration proves that no pattern exists that covers more faults from F_{ND}. Thus, the list of currently modeled faults F_{ND} and the global fault list F are pruned by fault simulation. The pattern p_j is added to the pattern sequence P and the bit-flip constraint $\Phi_{B_{numBF}}$ is reset (l. 16-18). When more than 80% of the faults contained in F_{ND} are detected, the model is rebuilt with the next *maxF* undetected faults (l. 20).

In the second case, no pattern was found (l. 22) and the iteration proves that no pattern exists that detects even a single fault under the constrained amount of bit-flips. Thus, *numBF* is increased and the for-loop is re-executed, thereby ensuring that the next pattern is allowed to spend more bit-flips during the fault coverage maximization.

Algorithm 1 Iterative Bit-Flipping Scan Test Pattern Generation.

```
1:  function GENERATEBFSPATTERNS(P, j, F, maxF)
2:      F_ND ← getND(F, maxF)                              ▷ get maxF not detected faults
3:      Φ ← Φ_{C_C} ∧ Φ_{F_ND} ∧ Φ_B                       ▷ build model
4:      numBF ← 1; update(Φ_{B_numBF})                     ▷ allow 1 bit-flip per register
5:      while F ≠ ∅ do
6:          for numF ← 1, ··· , maxF do                    ▷ cover maximum faults
7:              update(Φ_{F_numF})                         ▷ update cardinality constraint
8:              SAT ← solveΦ(Φ_{p_{j−1}}, Φ_{B_numBF}, Φ_{F_numF})   ▷ under assumptions
9:              if SAT then                                ▷ remember found pattern
10:                 p_j ← extractPattern(Φ); pFound ← true
11:             else
12:                 break for-loop                         ▷ (line 6)
13:             end if
14:         end for
15:         if pFound then                                 ▷ accept found pattern
16:             F ← fsim(p_j, F); F_ND ← fsim(p_j, F_ND)   ▷ prune fault lists
17:             P ← P ∪ p_j; j ← j + 1; pFound ← false     ▷ add pattern p_j
18:             numBF ← 1; update(Φ_{B_numBF})             ▷ reset allowed bit-flips
19:             if |F_ND| < 0.2 · maxF then                ▷ update targeted faults
20:                 F_ND ← getND(F, maxF); update(Φ)
21:             end if
22:         else                                           ▷ no p exists under numBF
23:             numBF ← numBF + 1; update(Φ_{B_numBF})
24:         end if
25:     end while
26:     return P
27: end function
```

The surrounding while loop (l. 5-25) terminates when all faults from F are detected. As F only contains combinationally detectable faults and all inputs are fully controllable all faults will be detected, in the last resort by a pattern requiring a high amount of bit-flips. The model Φ is only modified when a sufficiently high number of targeted faults is covered. Otherwise, it is solved under the assumptions regarding the current sequential state, the number of allowed bit-flips and the number of to be covered faults. Hence, the use of incremental SAT solving is facilitated that reuses learned clauses identified in previous solver calls and thus accelerates the test generation.

9.4. Summary

The unified Bit-Flipping architecture from Chapter 8 supports test with a test access mechanism that allows the observation of compacted test responses and provides the ability to update the captured sequential state through incremental bit-flipping. The architecture can be used to apply arbitrary test sets that will directly profit from the implicit response compaction. However, no guarantees with respect to other relevant test metrics can be given and especially test time might be inadequate due to high amounts of bit-flips necessary to set up the next pattern.

To this end, the generation of test sequences that are able to profit from the capabilities offered by the unified Bit-Flipping architecture is depicted. The test sequence generation is mapped to a Boolean satisfiability problem which explicitly models the possibility to employ bit-flips of the captured sequential state to set up a subsequent pattern. The problem is solved incrementally by a heuristic that generates one test pattern of the sequence per iteration. The sequence generation is guided by cardinality constraints in order to minimize the amount of bit-flips required during pattern setup while maximizing the pattern's fault coverage.

Chapter 10
Experimental Evaluation of the Offline Test Scheme

Based on public and industrial circuits, the test access through infrastructure reuse (Section 8) and the associated test sequence generation (Section 9) are evaluated with respect to all relevant test metrics. These include the area overhead (Section 10.2), the test time (Section 10.3), the test volume (Section 10.4), as well as the peak and average test power (Section 10.5), and the test energy (Section 10.6). Appendix A provides details on the employed design automation flow (Section A.1), the properties of the used benchmark circuits (Section A.2), and extended results for more circuits in tabulated form (Section A.4).

10.1. Experimental Setup

The hardware overhead of the unified architecture from Section 8.1 is evaluated for single registers of different sizes and benchmark circuits. These include public circuits from the ISCAS89 and ITC99 benchmark collections [BBK89; Dav99; CRS00] as well as industrial circuits kindly provided by NXP (formerly Philips). For each circuit, a combinational circuit representation is generated by substituting pseudo-primary in- and outputs for all sequential elements. This combinational core C_C is then mapped to the 45 nm Nangate Open Cell Library [Nan11]. [1]

As the benchmark circuits target an *edge-triggered design style*, the level-sensitive Bit-Flipping Latch (BFL) from Chapter 5 cannot directly be used to implement the low-level correction within the sequential circuit portion. However, it can easily

[1] Mapping of the combinational core to the target library preserves the original circuit structure. Synthesis as performed in [IW14] aims to optimize the area while preserving the circuit behavior.

be extended into a Bit-Flipping Flip-Flop (BFFF) by abutment of a standard master latch as shown in Figure 10.1. The Bit-Flipping Flip-Flop has similar properties as the Bit-Flipping Latch and has a standard cell area of 5.054 μm^2.

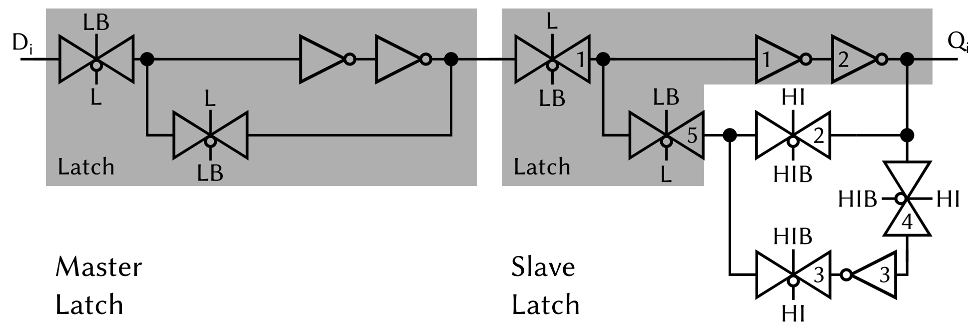

Figure 10.1.: Schematic of the Bit-Flipping Flip-Flop (BFFF).

The combinational core C_C is then combined with five different sequential payloads:

- a) Original: D-Flip-Flops (DFF, 4.522 μm^2).
- b) Scan Design: Scannable D-Flip-Flops (SDFFR, 6.916 μm^2).
- c) Scan Design + Fault Tolerance (FTScan): Scannable D-Flip-Flops together with a bitwise fault tolerance scheme comparable to RAZOR [EKD+03] or GRAAL [Nic07]: A shadow latch (DLH, 2.926 μm^2), an exclusive OR (XOR2, 1.596 μm^2) and a multiplexer (MUX2, 1.862 μm^2).
- d) Bit-Flipping Scan (BFScan): Bit-Flipping Flip-Flops (BFFF, 5.054 μm^2) combined with the characteristic computation and the syndrome decoder as well as scannable characteristic registers (SDFFR, 6.916 μm^2) (see Figure 8.1).
- e) Area Efficient Bit-Flipping Scan (BFScan TG): Bit-Flipping Scan with the Transmission-Gate Exclusive OR (TGXOR_X1, 0.798 μm^2) from Chapter 7 used in the characteristic tree, its parity protection and the syndrome comparator.

For b) and c), all FFs are organized into scan chains with a maximum length of 127. For d) and e), a register is implemented for each chain from the scan chain configuration used in b) and c). Note, that a chain length of 127 is very short. For longer chains, the scan based payloads will scale linear in terms of area and test time. The area of

the unified architecture and the test time of Bit-Flipping Scan sequences will grow slower due to the logarithmic correlation between n and l (see Eq. (4.6)).

The behavior during test application is evaluated for public and industrial benchmark circuits equipped with two of the above mentioned architectures. In particular, these include the 'FTScan' architecture, a combination of classical scan design and fault tolerance as a representative for a bitwise implementation, as well as the 'BFScan' architecture, as a representative for the presented unified architecture relying on a logarithmic checksum combined with bit-flipping.

To compare the test specific attributes, test sets for the former architecture are generated by a state-of-the-art test pattern generation tool respectively the presented test sequence generation algorithm for the latter architecture. The Bit-Flipping Scan test sequence generation discussed in the last chapter is implemented as a module within an inhouse Electronic Design Automation framework written in Java. The full featured Boolean reasoning library 'Sat4j' [LP10], a Java implementation based on MiniSAT [ES04] that supports native cardinality constraints, is used to build and solve the described models. Test conduction is simulated by applying the generated test sequences via a test bench. During simulation, the test bench records the number of cycles used (test time), the amount of bits exchanged over the test access mechanism (test volume) as well as the switching activity. The switching activity is finally used to perform a cycle accurate power analysis and determine the peak and average test power as well as the test energy.

10.2. Area Overhead

The different payloads are now evaluated and compared with respect to their area overhead. To this end, the overhead is first highlighted as a function of the register size before applying the sequential payloads to benchmark circuits which will be used later to quantify the quality of the provided test access.

10.2.1. Dependence on Register Size

The area overhead for different register sizes is depicted in Figure 10.2 where all results are normalized to the 'Original' payload area to ease comparability. The first two payloads have a constant area overhead, which amounts to around +50 % of the

original register size for classical scan design ('Scan') and slightly more than +200 % if scan design and bitwise fault tolerance are combined ('FTScan').

The unified architecture has an elevated overhead for very small registers which rapidly declines with growing register size ('BFScan'). For a 15-bit register, the overhead is already lower than for 'FTScan' and amounts only +131.91 % for a 127-bit register. A further reduction is achieved when the area efficient Transmission-Gate XOR cell is used ('BFScan TG'). For registers with at least 127 bits test access and fault tolerance as in the classical 'FTScan' architecture are provided by the 'Area Efficient Bit-Flipping Scan' architecture with a bisected area overhead.

10.2.2. Application to Benchmark Circuits

The results for the application to benchmark circuits are provided in Figure 10.3. Compared to the results regarding only a single register, the area overhead now depends on the ratio between the sequential and the combinational circuit part. The circuits from the ISCAS89 suite (starting with 's') tend to contain more sequential logic in contrast to the newer circuits from the ITC99 suite (denoted by 'b'). Scan design introduces an overhead between +5.2 % ('b21') and +17.5 % ('s35932') ('Scan'). In

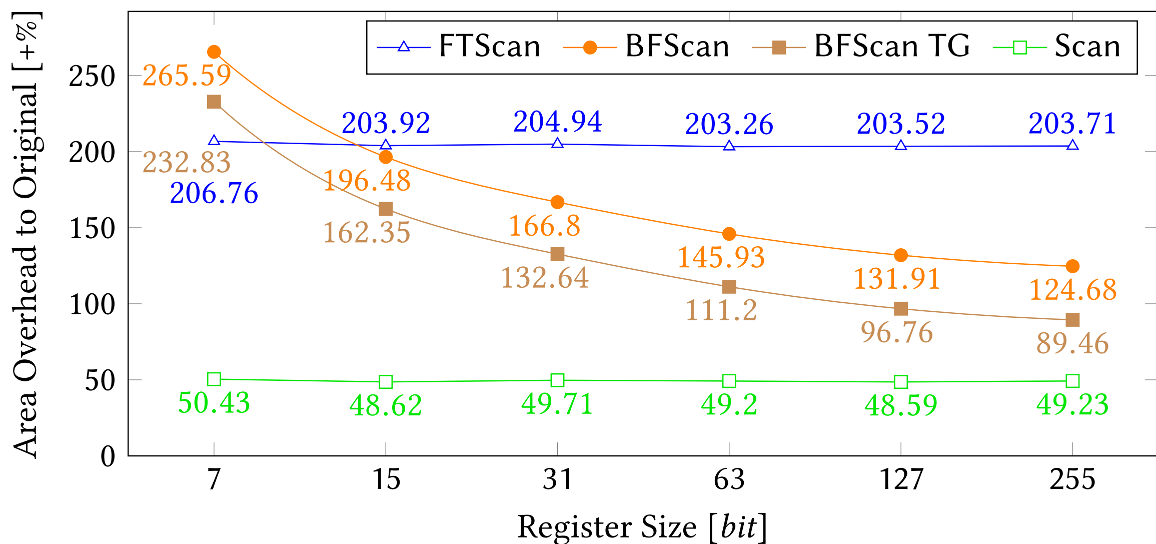

Figure 10.2.: Unified Architecture - Area Overhead for a Single Register (data from Table A.13).

combination with the orthogonal implementation of bitwise fault tolerance ('FTScan') the overhead is increased to a range between +22.4 % ('b21') and +75.6 % ('s35932').

The presented unified architecture inherits a lower area overhead for all circuits ('BFScan') of at least +14.5 % ('b21') and at most +48.4 % ('s35932'). Compared to the reference area ('FTScan') the overhead is lowered by +7.9 % ('b21') in the worst and +27.2 % ('s35932') in the best case. The use of the area optimized variant ('BFScan TG') continues to reduce the overhead and consistently results in an area overhead bisection for all circuits.

The results for industrial benchmark circuits (starting with 'p') are more counterbalanced due to a higher sequential fraction except for circuit 'p469k', which contains an extraordinary low amount of sequential elements. With this circuit excluded, the overhead of 'FTScan' is between +30.9 % ('p81k') and +54.6 % ('p239k'). By using the 'Area Efficient Bit-Flipping Scan' architecture the overhead is lowered to +15 % respectively +26.4 % and thereby reduced by more than half.

In summary, the results show that the unified architecture targeting both, test and fault tolerance, incorporates a significantly lower area overhead than the orthogonal combination of the two classical methods.

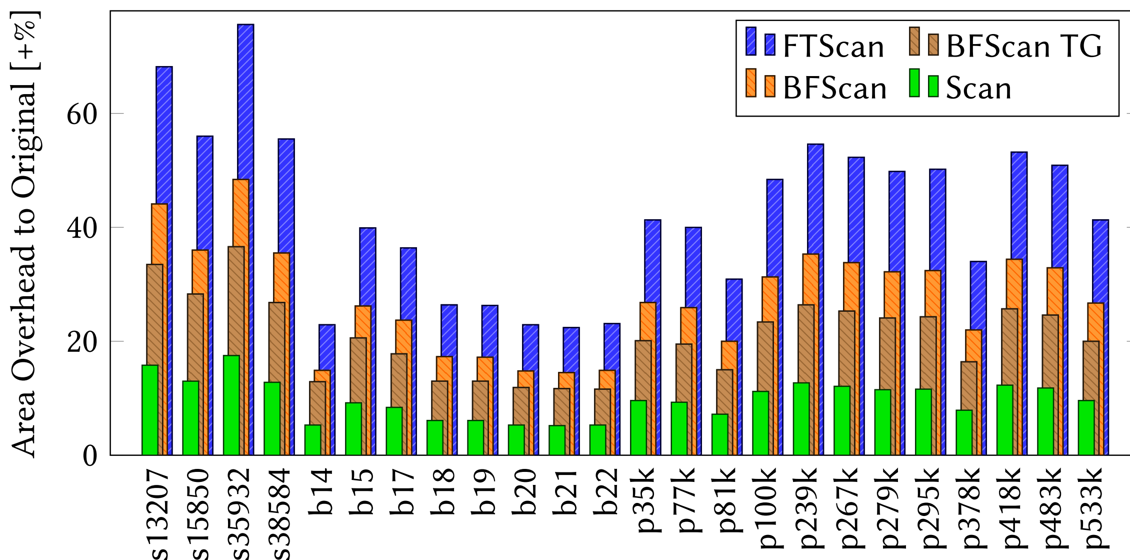

Figure 10.3.: Unified Architecture - Area Overhead for Benchmark Circuits (data and additional results in Table A.14).

10.3. Test Application Time

To measure and compare the test application time (TAT) a highly compacted test set is generated for each 'FTScan' configuration using a commercial ATPG. The heuristic from Section 9.3 is used to generate BFScan test sequences, where *maxF* is set to 500, providing a good tradeoff between runtime and achieved TAT. [2]

The achieved test application times are partially depicted in Figure 10.4 and are normalized to the test time required by scan design. For Bit-Flipping Scan the test application is faster for all circuits and a minimum speedup of at least 2.86 X is observed ('b18'). For a majority of the circuits, speedups of more than 5 X are achieved with a maximum speedup of 9.44 X ('p35k'). The not visualized results provided additionally in Tables A.16 and A.17 confirm and even outperform these observations. Test time is consistently reduced in all cases and many circuits exhibit speedups in excess of 5 X with a maximum speedup of 11.73 X for circuit 'p45k'.

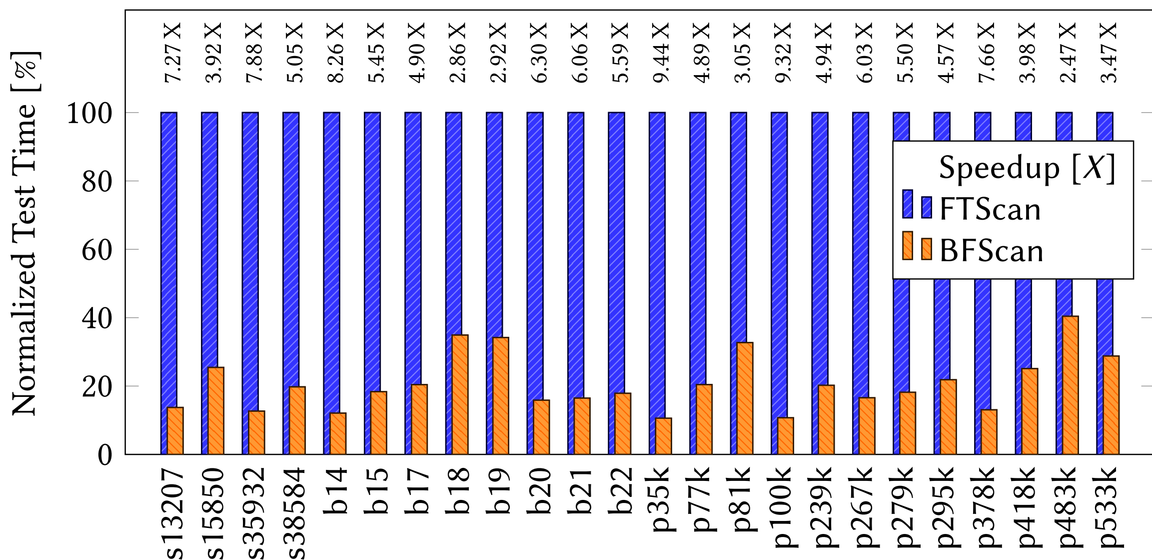

Figure 10.4.: Test Application Time for Benchmark Circuits (data and more results in Tables A.16 and A.17).

[2] Due to sufficient degrees of freedom in choosing a large subset of faults covered by the next pattern and excessive reuse of clauses learned in previous solver calls during incremental SAT solving. [IW14] used *maxF* = 100 and rebuilt the model whenever 10 % of F_{ND} were covered.

The absolute values provided in Tables A.16 and A.17 unveil that although the test time is significantly lowered by Bit-Flipping Scan (col. 10 and 5) in fact more patterns are contained in the sequential BFScan test sequence than in the combinational test set for scan design (col. 6 and 2). Instead of n shift cycles per pattern, only $l = \lceil log_2(n) \rceil$ cycles are required per pattern and bit-flip (col. 4 and 9), thus allowing to apply more patterns in shorter time. While this seems to indicate a raised coverage of non-target faults, further investigation is needed. Finally, the quotient of col. 8 and 6 indicates, that BFScan on average requires only few bit-flips to set up the following pattern.

In summary, in addition to the reduced area overhead of the unified Bit-Flipping Scan architecture, the achieved test application time speedups substantiate the effectiveness and efficiency of the corresponding test sequence generation. In addition, the shortened test times leverage a test cost reduction by shortening the allocation of expensive automatic test equipment per device under test.

10.4. Test Data Volume

The test data volume visualized in Figure 10.5 includes all bits exchanged with the circuit over primary and pseudo-primary in- and outputs during test application.

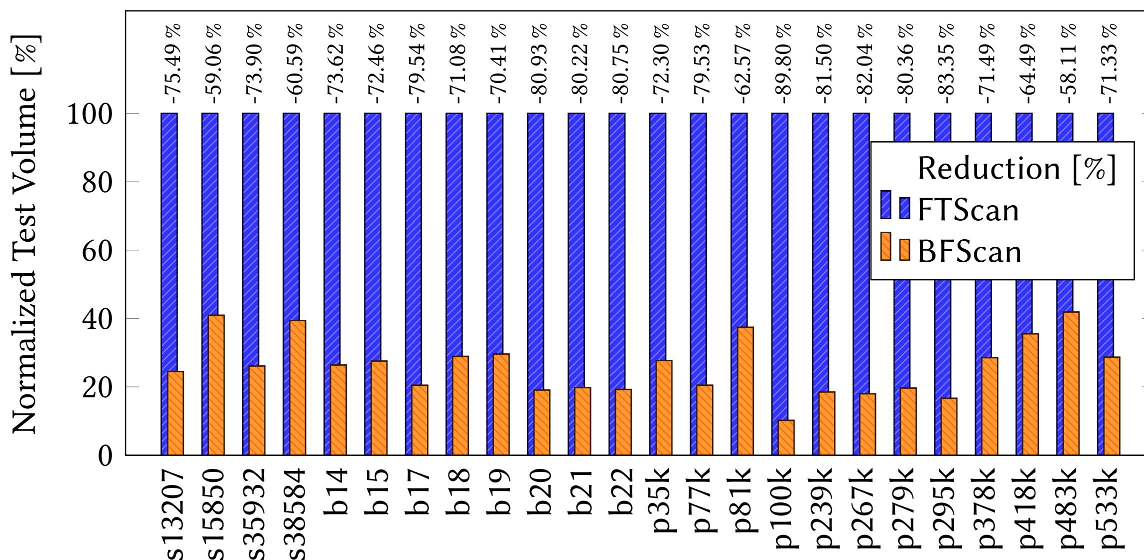

Figure 10.5.: Test Data Volume for Benchmark Circuits (data and more results in Tables A.16 and A.17).

The test data volume is reduced by Bit-Flipping Scan for all circuits. For circuit 'b14', the test data for BFScan amounts just 26.38 % of the test data required by classical scan design. Thus, the application of Bit-Flipping Scan entails a significant test data volume reduction by 73.62 %. For the majority of the examined public and industrial circuits, Bit-Flipping Scan exhibits a test data volume reduction of more than 70 %.

In summary, the application of Bit-Flipping Scan significantly reduces the memory requirements posed on the automatic test equipment. Hence, the use of less sophisticated and thus cheaper test equipment as well as a prolonged application of equipment before its decommissioning contribute towards a further test cost reduction.

10.5. Peak and Average Test Power

The power consumption of a chip during operation has excessive influence in two areas. First, careful dimensioning of the chip internal power distribution network to avoid the occurrence of power droop and voltage sag during *peak power* consumption. Second, proper dimensioning of the package and heat sink to dissipate the heat produced by the *average power* consumed. Thus, difficulties arise during test as the goal of exercising all fault locations in a short time period raises the switching activity and violates the peak and average power bounds determined by normal operation.

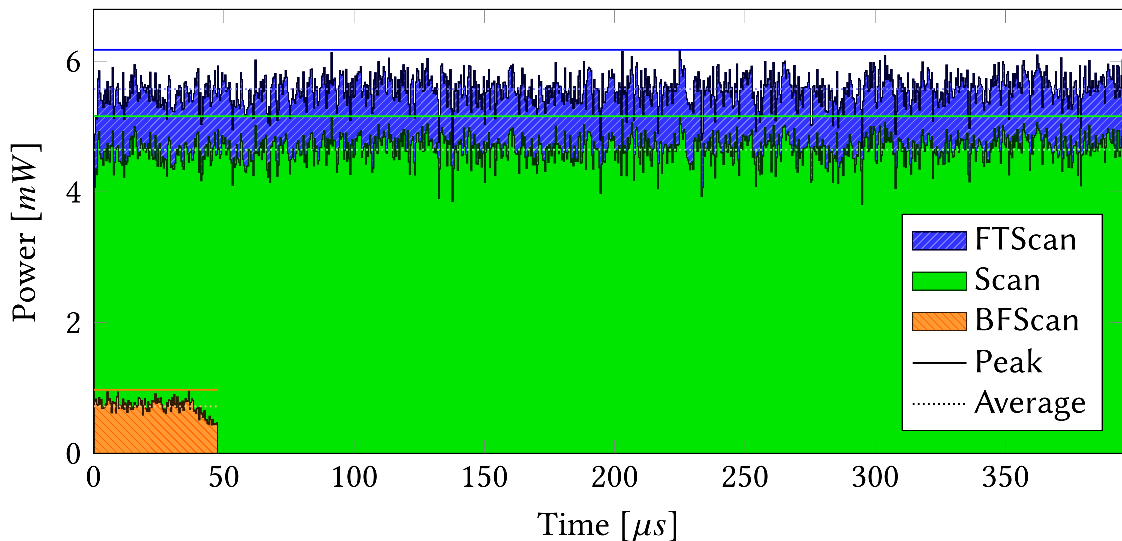

Figure 10.6.: Peak and Average Test Power and Test Energy for Circuit 'b14'.

Figure 10.6 depicts the power consumption of circuit 'b14' for three of the previously introduced architectures. During the application of the reference test set for classical scan design ('Scan') an average power of 4.647 mW is consumed with a peak power consumption of 5.156 mW at 224.8 μs. The combination of scan design with bitwise fault tolerance ('FTScan') uses the same test set but has a higher average power consumption of 5.570 mW and a peak power of 6.176 mW (at 224.8 μs) due to the shadow latches and exclusive OR gates included for error detection. The use of Bit-Flipping Scan ('BFScan') considerably lowers the average power to 0.7178 mW and the peak power to 0.9743 mW (at 36.4 μs).

The peak and average power for the used public and industrial circuits is depicted in Figures 10.7 and 10.8. For the public circuits, the peak power is reduced by between 84.22 % ('b14') and 89.55 % ('b19a') while the average power reduction ranges between 86.75 % ('s35932') and 90.71 % ('b17a'). For the industrial circuits, savings in a range from 77.79 % ('p469k') to 89.89 % ('p295k') are observed for the peak power and the average power is lowered by between 82.78 % ('p469k') and 91.02 % ('p500k').

In summary, along with the previously discussed advantages in terms of test time and test volume, Bit-Flipping Scan is a viable test solution that inherits considerable benefits in limiting the peak and reducing the average test power consumption.

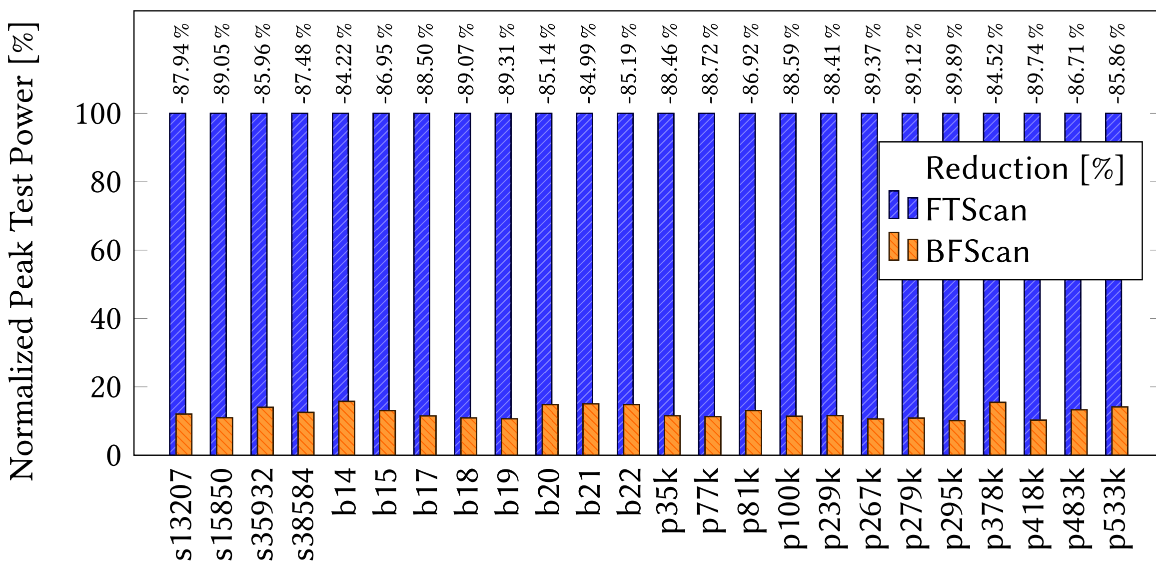

Figure 10.7.: Peak Test Power for Benchmark Circuits (data and more results in Tables A.18 and A.19).

10.6. Test Energy

The last sections showed, that the application of Bit-Flipping Scan inherits a lowered average test power consumption which is also consumed for much shorter time periods due to the reduced test application time. Thus, *test energy*, the product of power consumption and time, profits from a reduction in both dimensions.

The test energy for circuit 'b14' is equal to the area in Figure 10.6. The reference test set used for classical scan design ('Scan') and its combination with bitwise fault tolerance ('FTScan') both have a test application time of 395260 ns (98815 cycles from Table A.16 multiplied with a clock period of 4 ns due to a used test frequency of 250 MHz). Thus, the test energy of classical scan design amounts $1.83 \cdot 10^3$ nJ ('Scan') while the orthogonal combination of scan design and fault tolerance has an energy of $2.20 \cdot 10^3$ nJ ('FTScan'). In contrast, the test energy of Bit-Flipping Scan is considerably lower and constitutes with $3.43 \cdot 10^1$ nJ a reduction by 98.44 % compared to FTScan.

The test energy for the FTScan and BFScan architectures is depicted for all public and industrial circuits with logarithmic scale in Figure 10.9. For the public circuits the test energy is reduced by at least 96.64 % ('b18') while the maximum reduction of 98.55 % is achieved for circuit 's13207'. The industrial benchmarks show energy

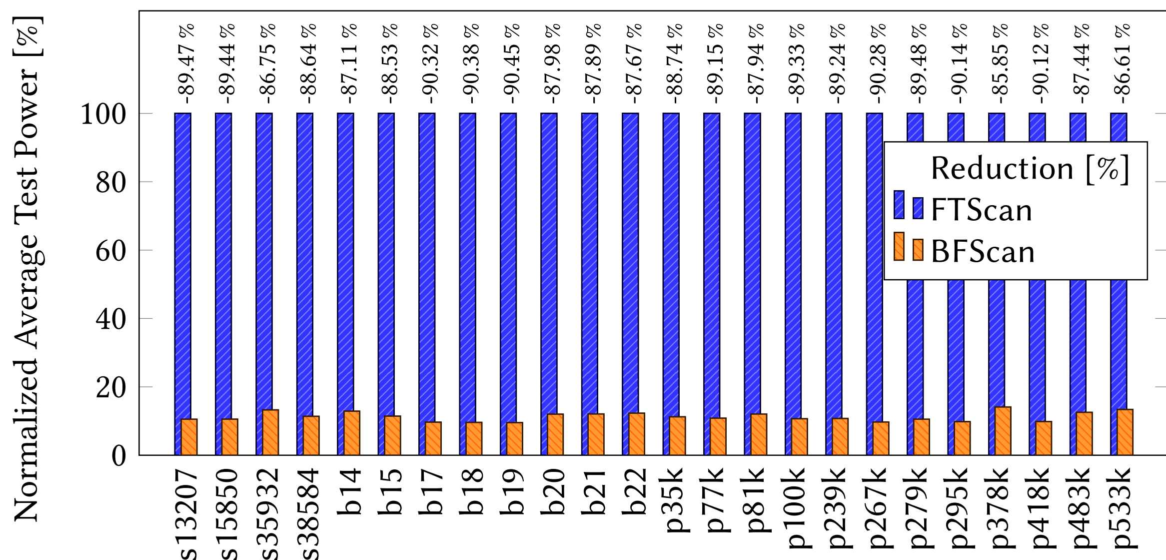

Figure 10.8.: Average Test Power for Benchmark Circuits (data and more results in Tables A.18 and A.19).

savings in the same order between 94.92 % ('p483k') and 98.97 % ('p45k'). In most cases, test energy is reduced by more than 97 %.

In summary, Bit-Flipping Scan positively influences both test power dimensions. Hence, the use of Bit-Flipping scan consistently lowers the test energy for all circuits by more than one order of magnitude. The absence of outliers with a low reduction as well as the narrowness of all observed reduction ratios further fortifies the efficiency of the Bit-Flipping Scan architecture and the associated test sequence generation.

10.7. Summary

For small registers, the hardware overhead associated with the unified architecture is already comparable to the overhead of an orthogonal combination of bitwise fault tolerance with scan design. With growing register sizes, the overhead declines logarithmically and is reduced by more than one third for large registers. By incorporating the area efficient *Transmission-Gate Exclusive OR* standard cell from Chapter 7 the area overhead is even further reduced and amounts only half of the overhead that comes along with a orthogonal bitwise implementation. The results for public

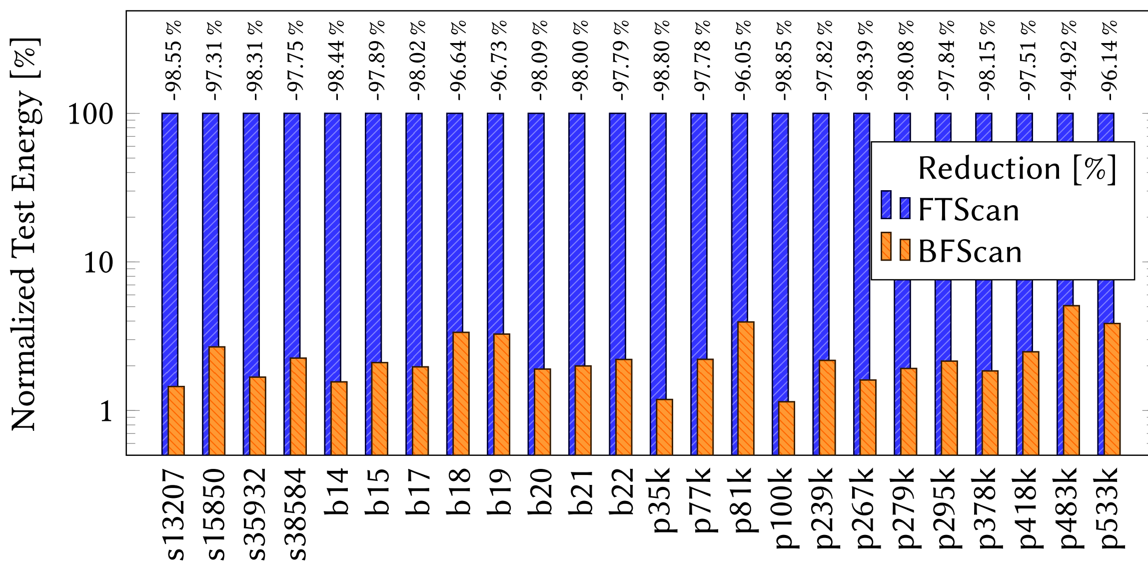

Figure 10.9.: Test Energy for Benchmark Circuits (logarithmic scale, data and more results in Tables A.18 and A.19).

benchmark and industrial circuits confirm the bisected area overhead for all circuits and demonstrates the applicability and viability of Bit-Flipping Scan as a unified architecture that supports test and fault tolerance.

The generated Bit-Flipping Scan test sequences are evaluated and compared to state-of-the-art test sets applied through scan design for public and industrial circuits with respect to all relevant test metrics. The results express preferences of the unified architecture in terms of test application time with a speedup of at least 5 X and test data volume which is reduced by more than 70 % for a majority of the circuits. In addition, peak and average test power are reduced by around 90 % and test energy is lowered by more than 95 %.

Thus, Bit-Flipping scan leverages the use of cheaper automatic test equipment due to significantly lowered requirements on the tester memory (test data volume reduction) and the tester power supply system (peak test power reduction). Furthermore, the test equipment allocation is shortened per device under test (test application time reduction) and device handling is eased due to relaxed cooling requirements (average test power reduction). In addition, even more complex tests might be facilitated during waver level test, where no external cooling is available and the thermal capacitance of the waver needs to be utilized (test energy reduction). Hence, the application of Bit-Flipping Scan contributes towards test cost reduction through manifold mechanisms without sacrificing test quality.

Summary and Discussion of Part III

This part targeted hard faults by offline testing. Proving the presence of hard faults since ever has been a necessity for every produced chip and is typically supported by dedicated test infrastructure in order to increase testability and attenuate test cost. Instead of implementing classical scan design in addition to bitwise fault tolerance to cope with hard faults and soft errors, a novel unified architecture is developed that provides fault tolerance concurrent to operation and supports test with an effective test access mechanism otherwise. While being able to support arbitrary test sequences, the full amenities of the architecture are unlocked by a test generation heuristic that takes advantage of the provided capabilities.

Chapter 8 - Test Access through Infrastructure Reuse - extends the fault tolerance architecture depicted in the last part to also serve as a test access mechanism. During test, reaching a high testability is crucial and mandates for full accessability to the sequential circuit state. Here, test access is provided with limited area overhead by repurposing the two fundamental concepts used during fault tolerance. The *modulo-2 address characteristic* of a register allows to observe the test response in compacted form. The *bit-flipping* mechanism permits to set up the next test pattern from the last test response by an incremental update of crucial register bits. As both, the characteristic and the bit-flip addresses, depend logarithmically on the register size, the 'Bit-Flipping Scan' architecture provides a high test access efficiency as long as the number of accesses per test pattern can be limited.

Chapter 9 - Test Sequence Generation - maps the test sequence generation to a Boolean satisfiability problem. The used model adheres the aspects necessary for sequential test generation: The combinational circuit core, the representation of faults and the sequential mapping between consecutive timeframes with the unique possibility to alter the sequential state between two timeframes by bit-flipping. While finding an optimal test sequence with a minimal test time for a given upper bound on the number of allowed bit-flips is shown to be possible in theory, it is only feasible for rather small model sizes due to the high complexity of sequential ATPG. Instead, the

satisfiability problem is solved heuristically under two cardinality constraints that allow to adjust the maximum amount of allowed bit-flips and the minimum amount of to be covered faults. Thereby, each pattern of the iteratively generated sequence is guaranteed to use a minimized amount of bit-flips while covering a maximized subset of the still undetected faults.

Chapter 10 - Experimental Evaluation of the Offline Test Scheme - finally reviews the application of the infrastructure reuse to benchmark circuits with respect to all relevant test metrics. The area overhead of the unified architecture is compared to the overhead required by the orthogonal combination of scan design with bitwise fault tolerance. For small registers the overhead is in the same range and reduces with growing register sizes. For large registers the constant overhead of the orthogonal combination is undercut by one third. A further optimization is achieved by applying the Transmission-Gate Exclusive OR from Chapter 7 that implements the XOR function with half of the area. The evaluation for public and industrial circuits affirms the bisected area overhead for all circuits and substantiates the efficacy and efficiency of the unified architecture to assist test and fault tolerance. Compared to test sequences targeting the orthogonal combination of scan design with bitwise fault tolerance, the test sequences tailored to the unified architecture are shown to result in a test time speedup of at least $5\,X$, reduce test volume by 70 %, limit peak test power to 10 %, lower average test power by an order of magnitude and decrease test energy by more than 95 %.

Chapter 11
Conclusions

The ability to escalate the integration density of digital circuits throughout the last decades finally culminates in their omnipresence in everyday life. This progressive proliferation also comprehends more and more safety critical application areas where failure is not an option. Thus, reliability, the probability of survival beyond a specified time, evolves as a primary objective. However, with an end of scaling in sight, reaching and maintaining a high reliability is severely challenged as every further scaling step raises the probability of failure. Hard faults unavoidable during manufacturing endanger the yield of fully functional chips and the elevated susceptibility to radiation-induced soft errors degrades the reliability during operation.

Consequently, every produced chip undergoes testing to screen the presence of hard faults throughout the manufacturing process which is supported by additional on-chip infrastructure for test access. Throughout the operation, fault tolerance infrastructure is used to mitigate the effect of soft errors with the help of redundancy. Due to the seemingly contradicting goals of test, that shows the presence of hard faults, and fault tolerance, that hides the occurrence of soft errors, both causes of failure are often contemplated independently. Thus, hard faults and soft errors are treated by two specialized but distinct infrastructures that are implemented orthogonally at elevated cost although they are never used at the same time.

The work at hand supports both application areas through a unified architecture with a reduced infrastructure area footprint that allows to provision fault tolerance during operation while acting as a test access mechanism during test. To this end, the unified architecture employs a unique combination of an efficient sequential state checksum with an effective state update by bit-flipping.

During fault tolerance, each individual register is protected by a combination of information and structural redundancy. The register specific error condition in form

of a modulo-2 address characteristic is derived area efficiently by a combinational characteristic tree with optimal organization. False detections are completely avoided by protecting the stored error condition. In contrast to classical bitwise fault tolerance, Single Event Upset detection is achieved with a considerably lower area overhead. By using the localization information compromised in the characteristic to control the bit-flipping capability embedded in new standard cells at low level a self-contained and fast concurrent online correction of single bit upsets is achieved. In presence of multiple bit upsets, the characteristic is easily extended without sacrificing efficiency in order to classify upsets according to their multiplicity and identify correctable Single Bit Upsets. As the checksum computation solely relies on exclusive OR gates, a further area reduction is enabled by the presented area optimized XOR cell.

During test, the unified architecture acts as a test access mechanism where the logarithmic register checksums allow to observe inherently compacted test responses while the next test pattern is set up through a selective update of the captured sequential state by bit-flipping. The generation of test sequences that exploit the capabilities of the unified architecture is mapped to a Boolean satisfiability problem and solved heuristically. Cardinality constraints are used to guide the pattern generation and restrict the number of bit-flips while maximizing the fault coverage. The results for public and industrial benchmark circuits depict substantially accelerated test application times, considerable test volume reductions, peak test power limitations, significant average test power cutbacks, as well as remarkable test energy savings.

In summary, the unified architecture based on a unique combination of logarithmic register checksums with a low-level bit-flipping mechanism is practically applicable, inherits reduced infrastructure cost, and incorporates direct benefits in the fault tolerance and test domains.

11.1. Future Research Directions

The unified architecture depicted in this work provides fault tolerance by correcting single bit upsets during operation and supports test with the ability to flip individual register bits. However, if multiple bit-flips are necessary in a register a sequence of single bit-flips has to be used. Theoretically, *multiple bit-flips* can be performed in a *single clock cycle* as the bit-flipping mechanism is implemented per bit at low-level and the used characteristic can easily be extended to provide a higher Hamming distance

(see Chapter 6). Thus, a more sophisticated decoder could bridge the gap between an extended localization information and the excitation of multiple concurrent flips.

The application of the unified architecture as a test access infrastructure targets faults in the combinational circuit part. Although the infrastructure itself can be affected by production defects, a large subset of the associated *architecture-internal faults* is already implicitly covered. In order to improve the coverage of such faults that possibly affect the provided fault tolerance and test access, all infrastructure parts should be thoroughly exercised by *explicit test sequences*. The robust test sequence for XOR-trees from [THL89] could serve as a starting point for the characteristic tree.

While the work at hand focuses on external testing supported by automatic test equipment, the raising failure rates stemming from expedited infant mortality and aging demand for *Built-In Self Test (BIST)* performed throughout the lifetime. The presented test access mechanism is applicable to BIST and seems to be well suited due to its low area overhead, the reduced test data volume stored on-chip, the accelerated test application time as well as the lowered test power consumption and energy.

In order to support the test of *timing faults*, the discussed test access mechanism supports the application of two pattern tests required by fault models that consider the *temporal behavior* of a circuit, such as the transition, gate or path delay fault model. While common schemes are constricted to launch a transition 'on shift' or 'on capture' of the first test response, the presented infrastructure could facilitate a more flexible 'launch on bit-flip' with an appropriate bit-flipping aware test sequence.

Timing characterization allows to detect premature *aging* by means of online delay measurement enabled through double sampling. The infrastructure contains the functional register R as well as its characteristic register C which are controlled by two delayed clocks. Thus, with a known clock phase shift and a sweep over the clock frequency, it allows the determination of the lowest frequency that results in a mismatch between the stored reference and the recomputed characteristic.

Small delay defects might not yet violate the nominal circuit timing after manufacturing but point to underlying hardware marginalities which may degrade into *early life failures*. Their confirmation is impaired since the slack along sensitized paths is typically much larger than the defect size. *Faster-than-at-speed tests* allow their detection under raised clock frequencies. However, operating the circuit above the nominal frequency increases noise in the power and clock network, which threatens reliable detection and also causes over-testing. Thus, the significant peak and average

test power reduction of the Bit-Flipping Scan architecture eases faster-than-at-speed testing while the shortened test time help to obey a provided thermal budget.

The work at hand is not limited to Application Specific Integrated Circuits (ASICs). *Reconfigurable architectures* dynamically instantiate application specific accelerators during runtime to combine the high performance of hardware with the flexibility of software. However, the Field Programmable Gate Arrays (FPGAs) used as the underlying fabric are also susceptible to soft errors. While the configuration data that defines the fabric's behavior is by now protected through error correcting codes, soft errors in the sequential state of the implemented accelerators are not yet addressed [Xil14]. Hence, the presented unified architecture could conquer this issue by providing fault tolerance. It is especially suited for FPGAs as its two fundamental concepts, the characteristic computation and the bit-flipping mechanism, possess lightweight implementations. The bit-flipping functionality requires only one additional FPGA lookup table (LUT) in excess to the sequential element. The exclusive OR function that forms the base of the characteristic tree is not harder to implement by a LUT than any other gate function.

Dynamic reconfiguration also demands for appropriate testing of the underlying fabric prior to reconfiguration (pre-reconfiguration test, PRET) and the validation of the instantiated functionality after reconfiguration (post-reconfiguration test, PORT). While structural tests implemented by specialized test configurations are used during the PRET, PORT employs functional tests due to the lack of an appropriate test access mechanism. With the unified architecture provisioning fault tolerance during operation, the inherently contained test access mechanism could also facilitate the use of structural tests during PORT.

Bibliography

[AAS+10] R. Adiga, G. Arpit, V. Singh, K. Saluja, and A. Singh, "Modified T-Flip-Flop based scan cell for RAS", in *Proc. IEEE European Test Symposium (ETS)*, 2010, pp. 113–118. DOI: 10.1109/ETSYM.2010.5512773.

[AC95] V. Agrawal and S. Chakradhar, "Combinational ATPG theorems for identifying untestable faults in sequential circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 9, pp. 1155–1160, 1995. DOI: 10.1109/43.406717.

[ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 1, no. 1, pp. 11–33, 2004. DOI: 10.1109/TDSC.2004.2.

[AM15] J.-L. Autran and D. Munteanu, *Soft Errors: From Particles to Circuits*, ser. Devices, Circuits, and Systems. CRC Press, 2015, ISBN: 9781466590830.

[And80] H. Ando, "Testing VLSI with random access scan", *Proc. of Compcon*, pp. 50–52, 1980.

[ASW+07] O. Amusan, A. Steinberg, A. Witulski, B. Bhuva, J. Black, M. Baze, and L. Massengill, "Single Event Upsets in a 130 nm Hardened Latch Design Due to Charge Sharing", in *Proc. IEEE Intl. Reliability Physics Symposium*, Apr. 2007, pp. 306–311. DOI: 10.1109/RELPHY.2007.369908.

[BA00] M. Bushnell and V. Agrawal, *Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits*, ser. Frontiers in Electronic Testing. Kluwer Academic Publishers, 2000, ISBN: 978-0-7923-7991-1.

[Bau05] R. Baumann, "Soft errors in advanced computer systems", *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 258–266, May 2005. DOI: 10.1109/MDT.2005.69.

[Bau08] R. Baumann, in *Handbook of Semiconductor Manufacturing Technology, Second Edition*, R. Doering and Y. Nishi, Eds. CRC Press / Taylor & Francis Group, 2008, ch. 31: Effects of Terrestrial Radiation on Integrated Circuits, pp. 31.1–31.23, ISBN: 978-1-57444-675-3.

[BBK89] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits", in *Proc. IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, 1989, 1929–1934 vol.3. DOI: 10.1109/ISCAS.1989.100747.

[BC08] L. Breaux and S. Collins, in *Handbook of Semiconductor Manufacturing Technology, Second Edition*, R. Doering and Y. Nishi, Eds. CRC Press / Taylor & Francis Group, 2008, ch. 27: Yield Management, pp. 27.1–27.29, ISBN: 978-1-57444-675-3.

[BDS+11] D. Bull, S. Das, K. Shivashankar, G. Dasika, K. Flautner, and D. Blaauw, "A Power-Efficient 32 bit ARM Processor Using Timing-Error Detection and Correction for Transient-Error Tolerance and Adaptation to PVT Variation", *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 18–31, 2011. DOI: 10.1109/JSSC.2010.2079410.

[BHvMW09] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, Eds., *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, Feb. 2009, vol. 185, p. 980, ISBN: 978-1-58603-929-5.

[BKL+08] D. Blaauw, S. Kalaiselvan, K. Lai, W. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, "Razor II: In situ error detection and correction for PVT and SER tolerance", in *Proc. IEEE Intl. Solid-State Circuits Conf. (ISSCC)*, 2008, pp. 400–401, 622. DOI: 10.1109/ISSCC.2008.4523226.

[BNLC06] S. Boutobza, M. Nicolaidis, K. M. Lamara, and A. Costa, "A Transparent based Programmable Memory BIST", in *Proc. IEEE European Test Symposium (ETS)*, 2006, pp. 89–96. DOI: 10.1109/ETS.2006.7.

[Bor05] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation", *IEEE Micro*, vol. 25, no. 6, pp. 10–16, Nov. 2005. DOI: 10.1109/MM.2005.110.

[BS05a] D. H. Baik and K. K. Saluja, "Progressive random access scan: a simultaneous solution to test power, test data volume and test time", in *Proc. IEEE Intl. Test Conf. (ITC)*, 2005, pp. 1–10. DOI: 10.1109/TEST.2005.1583994.

[BS05b] D. Baik and K. Saluja, "State-reuse test generation for progressive random access scan: Solution to test power, application time and data size", in *Proc. IEEE Asian Test Symp. (ATS)*, 2005, pp. 272–277. DOI: 10.1109/ATS.2005.101.

[BSK04] D. H. Baik, K. K. Saluja, and S. Kajihara, "Random Access Scan: A solution to test power, test data volume and test time", in *Proc. IEEE Intl. Conf. on VLSI Design (VLSID)*, 2004, pp. 883–888. DOI: 10.1109/ICVD.2004.1261042.

[CK07] C. Chiang and J. Kawa, *Design for Manufacturability and Yield for Nano-Scale CMOS*. Springer, 2007, ISBN: 978-1-4020-5187-6.

[CNV96] T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron CMOS technology", *IEEE Trans. on Nuclear Science*, vol. 43, no. 6, pp. 2874–2878, 1996. DOI: 10.1109/23.556880.

[CO07] M. Chen and A. Orailoglu, "Improving Circuit Robustness with Cost-Effective Soft-Error-Tolerant Sequential Elements", in *Proc. IEEE Asian Test Symp. (ATS)*, 2007, pp. 307–312. DOI: 10.1109/ATS.2007.51.

[Con03] C. Constantinescu, "Trends and challenges in VLSI circuit reliability", *IEEE Micro*, vol. 23, no. 4, pp. 14–19, Jul. 2003. DOI: 10.1109/MM.2003.1225959.

[Coo71] S. A. Cook, "The Complexity of Theorem-Proving Procedures", in *Proc. ACM Symp. on Theory of Computing*, 1971, pp. 151–158. DOI: 10.1145/800157.805047.

[CRS00] F. Corno, M. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results", *IEEE Design & Test of Computers*, vol. 17, no. 3, pp. 44–53, 2000. DOI: 10.1109/54.867894.

[Czu13] A. Czutro, "Efficiency and Applications of SAT-Based Test Pattern Generation", PhD thesis, Albert-Ludwigs Universität Freiburg, 2013.

[Dav99] S. Davidson, "ITC'99 Benchmark Circuits - Preliminary Results", in *Proc. International Test Conference (ITC)*, 1999, pp. 1125–1125. DOI: 10.1109/TEST.1999.805857.

[DEFT09] R. Drechsler, S. Eggersglüß, G. Fey, and D. Tille, *Test Pattern Generation using Boolean Proof Engines*. Springer, 2009, ISBN: 978-90-481-2359-9.

[Del97] T. J. Dell, "A white paper on the benefits of chipkill-correct ECC for PC server main memory", 1997.

[DLL62] M. Davis, G. Logemann, and D. Loveland, "A Machine Program for Theorem-proving", *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, Jul. 1962. DOI: 10.1145/368273.368557.

[DOFR89] M. Damiani, P. Olivo, M. Favalli, and B. Ricco, "An analytical model for the aliasing probability in signature analysis testing", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 11, pp. 1133–1144, Nov. 1989. DOI: 10.1109/43.41499.

[DSS95] T. Damarla, C. Stroud, and A. Sathaye, "Multiple error detection and identification via signature analysis", *Journal of Electronic Testing (JETTA)*, vol. 7, no. 3, pp. 193–207, 1995. DOI: 10.1007/BF00995313.

[DTP+09] S. Das, C. Tokunaga, S. Pant, W. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw, "RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance", *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 32–48, 2009. DOI: 10.1109/JSSC.2008.2007145.

[EKD+03] D. Ernst, N. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation", *IEEE/ACM Intl. Symp. on Microarchitecture*, pp. 7–18, 2003. DOI: 10.1109/MICRO.2003.1253179.

[Eld59] R. D. Eldred, "Test routines based on symbolic logical statements", *Journal of the ACM (JACM)*, vol. 6, no. 1, pp. 33–37, 1959.

[EMA11] M. Ebrahimi, S. Miremadi, and H. Asadi, "ScTMR: A scan chain-based error recovery technique for TMR systems in safety-critical applications", in *Proc. Design, Automation Test in Europe (DATE)*, 2011, pp. 1–4. DOI: 10.1109/DATE.2011.5763277.

[EMAF13] M. Ebrahimi, S. Miremadi, H. Asadi, and M. Fazeli, "Low-Cost Scan-Chain-Based Technique to Recover Multiple Errors in TMR Systems", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1454–1468, Aug. 2013. DOI: 10.1109/TVLSI.2012.2213102.

[ES03] N. Eén and N. Sörensson, "Temporal induction by Incremental SAT Solving", *Electronic Notes in Theoretical Computer Science*, vol. 89, no. 4, pp. 543–560, 2003. DOI: 10.1016/S1571-0661(05)82542-3.

[ES04] N. Eén and N. Sörensson, "An Extensible SAT-solver", in *Theory and Applications of Satisfiability Testing*, ser. Lecture Notes in Computer Science, E. Giunchiglia and A. Tacchella, Eds., vol. 2919, Springer, 2004, pp. 502–518. DOI: 10.1007/978-3-540-24605-3_37.

[EW77] E. B. Eichelberger and T. W. Williams, "A logic design structure for LSI testability", *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 462–468, 1977.

[EWI+08] M. Elm, H.-J. Wunderlich, M. E. Imhof, C. G. Zoellin, J. Leenstra, and N. Maeding, "Scan chain clustering for test power reduction", in *Proc. 45th ACM/IEEE Design Automation Conference (DAC)*, Anaheim, CA, USA, Jun. 2008, pp. 828–833. DOI: 10.1145/1391469.1391680.

[FFK+12] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. Harris, D. Blaauw, and D. Sylvester, "Bubble Razor: An architecture-independent approach to timing-error detection and correction", in *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*, 2012, pp. 488–490. DOI: 10.1109/ISSCC.2012.6177103.

[FFK+13] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. Harris, D. Blaauw, and D. Sylvester, "Bubble Razor: Eliminating Timing Margins in an ARM Cortex-M3 Processor in 45 nm CMOS Using Architecturally Independent Error Detection and Correction", *IEEE J. of Solid-State Circuits*, vol. 48, no. 1, pp. 66–81, Jan. 2013. DOI: 10.1109/JSSC.2012.2220912.

[FMEP09] M. Fazeli, S. Miremadi, A. Ejlali, and A. Patooghy, "Low energy single event upset/single event transient-tolerant latch for deep subMicron technologies", *IET Computers & Digital Techniques*, vol. 3, no. 3, pp. 289–303, 2009. DOI: 10.1049/iet-cdt.2008.0099.

[FPME07] M. Fazeli, A. Patooghy, S. Miremadi, and A. Ejlali, "Feedback Redundancy: A Power Efficient SEU-Tolerant Latch Design for Deep Sub-Micron Technologies", *Proc. of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 276–285, 2007. DOI: 10.1109/DSN.2007.51.

[GBMR06] A. Goel, S. Bhunia, H. Mahmoodi, and K. Roy, "Low-overhead design of soft-error-tolerant scan flip-flops with enhanced-scan capability", in *Proc. Asia and South Pacific Design Automation Conference*, 2006, pp. 1–6. DOI: 10.1109/ASPDAC.2006.1594762.

[GGR+04] M. Gordon, P. Goldhagen, K. Rodbell, T. Zabel, H. H. K. Tang, J. M. Clem, and P. Bailey, “Measurement of the flux and energy spectrum of cosmic-ray induced neutrons on the ground”, *IEEE Trans. on Nuclear Science*, vol. 51, no. 6, pp. 3427–3434, Dec. 2004. DOI: 10.1109/TNS.2004.839134.

[GJ79] M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman, 1979.

[GJD+14] N. Gaspard, S. Jagannathan, Z. Diggins, N. Mahatme, T. Loveless, B. Bhuva, L. Massengill, W. Holman, B. Narasimham, A. Oates, P. Marcoux, N. Tam, M. Vilchis, S.-J. Wen, R. Wong, and Y. Xu, “Soft error rate comparison of various hardened and non-hardened flip-flops at 28-nm node”, in *Proc. IEEE Intl. Reliability Physics Symposium*, Jun. 2014, SE.5.1–SE.5.5. DOI: 10.1109/IRPS.2014.6861177.

[GK83] D. Gajski and R. Kuhn, “Guest Editors’ Introduction: New VLSI Tools”, *IEEE Computer*, vol. 16, no. 12, pp. 11–14, Dec. 1983. DOI: 10.1109/MC.1983.1654264.

[GNR61] J. M. Galey, R. E. Norby, and J. Roth, “Techniques for the diagnosis of switching circuit failures”, in *Proc. Symp. on Switching Circuit Theory and Logical Design*, 1961, pp. 152–160. DOI: 10.1109/FOCS.1961.33.

[Gol79] L. Goldstein, “Controllability/observability analysis of digital circuits”, *IEEE Transactions on Circuits and Systems*, vol. 26, no. 9, pp. 685–693, Sep. 1979. DOI: 10.1109/TCS.1979.1084687.

[GSZ09] B. Gill, N. Seifert, and V. Zia, “Comparison of alpha-particle and neutron-induced combinational and sequential logic error rates at the 32nm technology node”, in *Proc. IEEE International Reliability Physics Symposium*, 2009, pp. 199–205. DOI: 10.1109/IRPS.2009.5173251.

[GT80] L. Goldstein and E. Thigpen, “SCOAP: Sandia Controllability / Observability Analysis Program”, in *Proc. IEEE/ACM Design Automation Conference (DAC)*, 1980, pp. 190–196. DOI: 10.1109/DAC.1980.1585245.

[Ham50] R. W. Hamming, “Error Correcting and Error Detecting Codes”, *Bell System Technical Journal*, vol. 29, pp. 147–160, 1950.

[HCB95] H. Hollander, B. S. Carlson, and T. D. Bennett, “Synthesis of SEU-tolerant ASICs using concurrent error correction”, in *Proc. IEEE Great Lakes Symposium on VLSI*, 1995, pp. 90–93.

[HKW+04] P. Hazucha, T. Karnik, S. Walstra, B. Bloechel, J. Tschanz, J. Maiz, K. Soumyanath, G. Dermer, S. Narendra, V. De, and S. Borkar, “Measurements and analysis of SER-tolerant latch in a 90-nm dual-VT CMOS process”, *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1536–1543, Sep. 2004. DOI: 10.1109/JSSC.2004.831449.

[HLH15] Z. Huang, H. Liang, and S. Hellebrand, “A High Performance SEU Tolerant Latch”, *Journal of Electronic Testing*, pp. 1–11, 2015. DOI: 10.1007/s10836-015-5533-5.

[HML+02] R. Hentschke, F. Marques, F. Lima, L. Carro, A. Susin, and R. Reis, “Analyzing area and performance penalty of protecting different digital modules with Hamming code and triple modular redundancy”, in *Proc. IEEE Symposium on Integrated Circuits and Systems Design*, 2002, pp. 95–100. DOI: 10.1109/SBCCI.2002.1137643.

[HNG+13] S. Hamdioui, M. Nicolaidis, D. Gizopoulos, A. Grasset, G. Guido, and P. Bonnot, “Reliability challenges of real-time systems in forthcoming technology nodes”, in *Proc. Design, Automation and Test in Europe (DATE)*, 2013, pp. 129–134. DOI: 10.7873/DATE.2013.040.

[HSS12] A. A. Hwang, I. A. Stefanovici, and B. Schroeder, “Cosmic Rays Don’t Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design”, in *Proc. ACM Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2012, pp. 111–122. DOI: 10.1145/2150976.2150989.

[Hua14] Z. Huang, “A high performance SEU-tolerant latch for nanoscale CMOS technology”, in *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2014, pp. 1–5. DOI: 10.7873/DATE.2014.175.

[HWI+02] S. Hellebrand, H.-J. Wunderlich, A. A. Ivaniuk, Y. V. Klimets, and V. N. Yarmolik, “Efficient Online and Offline Testing of Embedded DRAMs”, *IEEE Trans. on Computers*, vol. 51, no. 7, pp. 801–809, 2002. DOI: 10.1109/TC.2002.1017700.

[HWI+99] S. Hellebrand, H.-J. Wunderlich, A. Ivaniuk, Y. Klimets, and V. Yarmolik, “Error detecting refreshment for embedded DRAMs”, in *Proc. IEEE VLSI Test Symp. (VTS)*, 1999, pp. 384–390. DOI: 10.1109/VTEST.1999.766693.

[IS75] O. Ibarra and S. Sahni, “Polynomially Complete Fault Detection Problems”, *IEEE Trans. on Computers*, vol. C-24, no. 3, pp. 242–249, 1975. DOI: 10.1109/T-C.1975.224205.

[Ito90] N. Ito, “Automatic incorporation of on-chip testability circuits”, in *Proc. ACM/IEEE Design Automation Conference (DAC)*, 1990, pp. 529–534. DOI: 10.1109/DAC.1990.114912.

[ITR13] ITRS. (2013). International Technology Roadmap for Semiconductors, [Online]. Available: http://www.itrs.net/Links/2013ITRS/.

[IW11b] M. E. Imhof and H.-J. Wunderlich, “Korrektur transienter Fehler in eingebetteten Speicherelementen”, in *Proc. 5. GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf (ZuE)*, Hamburg-Harburg, Germany, Sep. 2011, pp. 76–83, ISBN: 978-3-8007-3357-6.

[IW11a] M. E. Imhof and H.-J. Wunderlich, “Soft Error Correction in Embedded Storage Elements”, in *Proc. 17th IEEE International On-Line Testing Symposium (IOLTS)*, Athens, Greece, Jul. 2011, pp. 169–174. DOI: 10.1109/IOLTS.2011.5993832.

[IW14] M. E. Imhof and H.-J. Wunderlich, “Bit-Flipping Scan - A Unified Architecture for Fault Tolerance and Offline Test”, in *Proc. Design, Automation and Test in Europe (DATE)*, Dresden, Germany, Mar. 2014, pp. 1–6. DOI: 10.7873/DATE.2014.206.

[IWZ08b] M. E. Imhof, H.-J. Wunderlich, and C. G. Zoellin, “Erkennung von transienten Fehlern in Schaltungen mit reduzierter Verlustleistung”, in *Proc. 2. GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf (ZuE)*, Ingolstadt, Germany, Sep. 2008, pp. 107–114, ISBN: 978-3-8007-3119-0.

[IWZ08a] M. E. Imhof, H.-J. Wunderlich, and C. G. Zoellin, “Integrating Scan Design and Soft Error Correction in Low-Power Applications”, in *Proc. 14th IEEE International On-Line Testing Symposium (IOLTS)*, Rhodes, Greece, Jul. 2008, pp. 59–64. DOI: 10.1109/IOLTS.2008.31.

[IZW+07] M. E. Imhof, C. G. Zoellin, H.-J. Wunderlich, N. Maeding, and J. Leenstra, “Scan Test Planning for Power Reduction”, in *Proc. 44th ACM/IEEE Design Automation Conference (DAC)*, San Diego, CA, USA, Jun. 2007, pp. 521–526. DOI: 10.1145/1278480.1278614.

[JED03] JEDEC, *Method for Developing Acceleration Models for Electronic Component Failure Mechanisms : JESD91A*. JEDEC Solid State Technology Association, Aug. 2003, pp. 1–20.

[JED06] JEDEC, *Measurement and Reporting of Alpha Particles and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices : JESD89A*. JEDEC Solid State Technology Association, Oct. 2006, pp. 1–94.

[KKF+13] S. Kim, I. Kwon, D. Fick, M. Kim, Y.-P. Chen, and D. Sylvester, "Razorlite: A side-channel error-detection register for timing-margin recovery in 45nm SOI CMOS", in *Proc. IEEE Intl. Solid-State Circuits Conf. (ISSCC)*, 2013, pp. 264–265. DOI: 10.1109/ISSCC.2013.6487728.

[KZIW08] M. A. Kochte, C. G. Zoellin, M. E. Imhof, and H.-J. Wunderlich, "Test Set Stripping Limiting the Maximum Number of Specified Bits", in *Proc. 4th IEEE International Symposium on Electronic Design, Test and Applications (DELTA)*, Best Paper Award, Hong Kong, China, Jan. 2008, pp. 581–586. DOI: 10.1109/DELTA.2008.64.

[Lap85] J. Laprie, "Dependable Computing and Fault Tolerance: Concepts and Terminology", in *Proc. IEEE Intl. Symp. on Fault-Tolerant Computing (FTCS)*, 1985, pp. 2–11. DOI: 10.1109/FTCSH.1995.532603.

[Lev73] L. A. Levin, "Universal sequential search problems", *Problems of Information Transmission*, vol. 9, no. 3, pp. 265–266, 1973.

[LHSC10] X. Li, M. C. Huang, K. Shen, and L. Chu, "A Realistic Evaluation of Memory Hardware Errors and Software System Susceptibility", in *Proc. USENIX Annual Technical Conference*, 2010, pp. 75–88.

[LKL11] S. Lin, Y.-B. Kim, and F. Lombardi, "Design and Performance Evaluation of Radiation Hardened Latches for Nanoscale CMOS", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 7, pp. 1315–1319, Jul. 2011. DOI: 10.1109/TVLSI.2010.2047954.

[LP10] D. Le Berre and A. Parrain, "The Sat4j library, release 2.2 system description", *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7, pp. 59–64, 2010.

[LSHC07] X. Li, K. Shen, M. C. Huang, and L. Chu, "A Memory Soft Error Measurement on Production Systems", in *Proc. USENIX Annual Technical Conference*, 2007, pp. 275–280.

[LV62] R. E. Lyons and W. Vanderkulk, "The Use of Triple Modular Redundancy to Improve Computer Reliability", *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962. DOI: 10.1147/rd.62.0200.

[MAS05b] A. S. Mudlapur, V. D. Agrawal, and A. D. Singh, "A novel random access scan flip-flop design", in *Proc. VLSI Design and Test Symp.*, 2005, pp. 11–13. DOI: 10.1.1.110.9077.

[MAS05a] A. S. Mudlapur, V. Agrawal, and A. D. Singh, "A random access scan architecture to reduce hardware overhead", in *Proc. IEEE Intl. Test Conf. (ITC)*, vol. 1, 2005, p. 350. DOI: 10.1109/TEST.2005.1583993.

[MB57] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits", in *Int. Symp. on Theory of Switching*, 1957, pp. 204–243.

[ME02] D. Mavis and P. Eaton, "Soft error rate mitigation techniques for modern microcircuits", in *Proc. IEEE Annual Reliability Physics Symposium*, 2002, pp. 216–225. DOI: 10.1109/RELPHY.2002.996639.

[Mea55] G. H. Mealy, "A Method for Synthesizing Sequential Circuits", *Bell System Technical Journal*, vol. 34, no. 5, pp. 1045–1079, 1955. DOI: 10.1002/j.1538-7305.1955.tb03788.x.

[MIY07] S. Musin, A. Ivaniuk, and V. Yarmolik, "Multiple Errors Detection Technique for RAM", in *Proc. IEEE Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, Apr. 2007, pp. 1–4. DOI: 10.1109/DDECS.2007.4295292.

[MK02] S. Mitra and K. S. Kim, "X-Compact: An Efficient Response Compaction Technique for Test Cost Reduction", in *Proc. IEEE Intl. Test Conf. (ITC)*, 2002, pp. 311–320. DOI: 10.1109/TEST.2002.1041774.

[MK03] S. Mitra and K. S. Kim, "XMAX: X-tolerant architecture for MAXimal test compression", in *Proc. IEEE International Conference on Computer Design (ICCD)*, 2003, pp. 326–330. DOI: 10.1109/ICCD.2003.1240914.

[MK04] S. Mitra and K. S. Kim, "X-Compact: An Efficient Response Compaction Technique", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 3, pp. 421–432, 2004. DOI: 10.1109/TCAD.2004.823341.

[MK06] S. Mitra and K. S. Kim, "XPAND: an efficient test stimulus compression technique", *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 163–173, Feb. 2006. DOI: 10.1109/TC.2006.31.

[Moo65] G. E. Moore, "Cramming more components onto integrated circuits", *Electronics*, vol. 38, no. 8, Apr. 1965.

[Moo75] G. E. Moore, "Progress in digital integrated electronics", in *Proc. IEEE International Electron Devices Meeting*, vol. 21, 1975, pp. 11–13.

[MSZ+05] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. Kim, "Robust system design with built-in soft-error resilience", *IEEE Computer*, vol. 38, no. 2, pp. 43–52, 2005. DOI: 10.1109/MC.2005.70.

[Mud01] T. Mudge, "Power: a first-class architectural design constraint", *IEEE Computer*, vol. 34, no. 4, pp. 52–58, Apr. 2001. DOI: 10.1109/2.917539.

[Muk08] S. Mukherjee, *Architecture design for soft errors*. Morgan Kaufmann Publishers/Elsevier, 2008, ISBN: 9780123695291.

[MWE+03] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor", in *Proc. IEEE/ACM International Symposium on Microarchitecture*, 2003, pp. 29–40. DOI: 10.1109/MICRO.2003.1253181.

[Nan11] Nangate. (Aug. 2011). Open Cell Library v1.3 v2010_12, [Online]. Available: http://www.si2.org/openeda.si2.org/projects/nangatelib.

[ND05] R. Naseer and J. Draper, "The DF-dice storage element for immunity to soft errors", in *Proc. of the 48th IEEE Intl. Midwest Symp. on Circuits and Systems*, 2005. DOI: 10.1109/MWSCAS.2005.1594099.

[Nic07] M. Nicolaidis, "GRAAL: a new fault tolerant design paradigm for mitigating the flaws of deep nanometric technologies", *Proc. IEEE Intl. Test Conf. (ITC)*, pp. 1–10, 2007. DOI: 10.1109/TEST.2007.4437666.

[Nic10] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*, ser. Frontiers in Electronic Testing. Springer Verlag, 2010, vol. 41, ISBN: 9781441969927.

[NII10] K. Namba, T. Ikeda, and H. Ito, "Construction of SEU Tolerant Flip-Flops Allowing Enhanced Scan Delay Fault Testing", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 9, pp. 1265–1276, Sep. 2010. DOI: 10.1109/TVLSI.2009.2022083.

[NP73] L. W. Nagel and D. O. Pederson, "SPICE (Simulation Program with Integrated Circuit Emphasis)", EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M382, Apr. 1973.

[NY03] H. Nguyen and Y. Yagil, "A systematic approach to SER estimation and solutions", in *Proc. IEEE Intl. Reliability Physics Symposium*, 2003, pp. 60–70. DOI: 10.1109/RELPHY.2003.1197722.

[OJC07] R. Oliveira, A. Jagirdar, and T. Chakraborty, "A TMR Scheme for SEU Mitigation in Scan Flip-Flops", in *Proc. IEEE Intl. Symp. on Quality Electronic Design (ISQED)*, 2007, pp. 905–910. DOI: 10.1109/ISQED.2007.25.

[PCK99] M. Prasad, P. Chong, and K. Keutzer, "Why is ATPG easy?", in *Proc. ACM/IEEE Design Automation Conference (DAC)*, 1999, pp. 22–28. DOI: 10.1109/DAC.1999.781224.

[PNL11] D. Palframan, K. N.S., and M. Lipasti, "Time Redundant Parity for Low-Cost Transient Error Detection", in *Proc. Design, Automation and Test in Europe (DATE)*, 2011, pp. 52–57. DOI: 10.1109/DATE.2011.5763017.

[RGF+03] P. Roche, G. Gasiot, K. Forbes, V. O'Sullivan, and V. Ferlet, "Comparisons of soft error rate for SRAMs in commercial SOI and bulk below the 130-nm technology node", *IEEE Trans. on Nuclear Science*, vol. 50, no. 6, pp. 2046–2054, Dec. 2003. DOI: 10.1109/TNS.2003.821588.

[RTK+02] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.-H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian, "Embedded deterministic test for low cost manufacturing test", in *Proc. IEEE Intl. Test Conf. (ITC)*, 2002, pp. 301–310. DOI: 10.1109/TEST.2002.1041773.

[RTKM04] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 776–792, 2004. DOI: 10.1109/TCAD.2004.826558.

[RTWR03] J. Rajski, J. Tyszer, C. Wang, and S. Reddy, "Convolutional compaction of test responses", in *Proc. IEEE Intl. Test Conf. (ITC)*, 2003, pp. 745–754. DOI: 10.1109/TEST.2003.1270904.

[RTWR05] J. Rajski, J. Tyszer, C. Wang, and S. Reddy, "Finite memory test response compactors for embedded test applications", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 622–634, Apr. 2005. DOI: 10.1109/TCAD.2005.844111.

[Rut72] R. Rutman, "Fault-detection test generation for sequential logic by Heuristic Tree Search", Hughes Aircraft Company, Ground Systems Division, Design Automation Group, Tech. Rep. TP-72-11-4, 1972, Was once available as IEEE Repository paper R-72-187, 1972.

[SABR04] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The impact of technology scaling on lifetime reliability", in *Proc. Intl. Conf. on Dependable Systems and Networks*, 2004, pp. 177–186. DOI: 10.1109/DSN.2004.1311888.

[SCW+07] J. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. Davis, P. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "FreePDK: An Open-Source Variation-Aware Design Kit", in *Proc. IEEE Intl. Conf. on Microelectronic Systems Education (MSE)*, 2007, pp. 173–174. DOI: 10.1109/MSE.2007.44.

[SEE98] M. Shams, J. Ebergen, and M. Elmasry, "Modeling and comparing CMOS implementations of the C-element", *IEEE Trans. on VLSI Systems*, vol. 6, no. 4, pp. 563–567, Dec. 1998. DOI: 10.1109/92.736128.

[Sex03] F. W. Sexton, "Destructive single-event effects in semiconductor devices and ICs", *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 603–621, Jun. 2003. DOI: 10.1109/TNS.2003.813137.

[SH04] J. Segura and C. F. Hawkins, *CMOS Electronics: How It Works, How It Fails.* IEEE Press Piscataway, 2004, ISBN: 978-0-471-47669-6.

[SKK+02] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic", in *Proc. Intl. Conf. on Dependable Systems and Networks (DSN)*, 2002, pp. 389–398. DOI: 10.1109/DSN.2002.1028924.

[SPW09] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM Errors in the Wild: A Large-scale Field Study", in *Proc. ACM Intl. Joint Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS)*, Seattle, WA, USA, 2009, pp. 193–204. DOI: 10.1145/1555349.1555372.

[ST04] N. Seifert and N. Tam, "Timing vulnerability factors of sequentials", *IEEE Trans. on Device and Materials Reliability*, vol. 4, no. 3, pp. 516–522, 2004. DOI: 10.1109/TDMR.2004.831993.

[THL+14] D. Tang, C. He, Y. Li, H. Zang, C. Xiong, and J. Zhang, “Soft error reliability in advanced CMOS technologies - trends and challenges”, *Science China Technological Sciences*, vol. 57, no. 9, pp. 1846–1857, 2014. DOI: 10.1007/s11431-014-5565-6.

[THL89] D. Tao, C. Hartmann, and P. Lala, “A robust test sequence for XOR trees”, in *Proc. IEEE Asilomar Conf. on Signals, Systems and Computers*, vol. 2, 1989, pp. 994–998. DOI: 10.1109/ACSSC.1989.1201047.

[Tou06] N. A. Touba, “Survey of test vector compression techniques”, *IEEE Design & Test of Computers*, vol. 23, no. 4, pp. 294–303, 2006. DOI: 10.1109/MDT.2006.105.

[Tse68] G. S. Tseitin, “On the Complexity of Derivation in Propositional Calculus”, *Studies in constructive mathematics and mathematical logic*, vol. 2, no. 115-125, pp. 10–13, 1968.

[Tse83] G. Tseitin, “On the Complexity of Derivation in Propositional Calculus”, in *Automation of Reasoning*, ser. Symbolic Computation, J. Siekmann and G. Wrightson, Eds., Springer, 1983, pp. 466–483. DOI: 10.1007/978-3-642-81955-1_28.

[vNeu56] J. von Neumann, “Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components”, in *Automata Studies*, C. E. Shannon and J. McCarthy, Eds., Princeton University Press, 1956.

[WA10] F. Wang and V. D. Agrawal, “Enhancing random access scan for soft error tolerance”, in *Proc. IEEE Southeastern Symp. on System Theory (SSST)*, 2010, pp. 263–268. DOI: 10.1109/SSST.2010.5442827.

[WA73] M. J. Y. Williams and J. B. Angell, “Enhancing Testability of Large-Scale Integrated Circuits via Test Points and Additional Logic”, *IEEE Trans. on Computers*, vol. 22, pp. 46–60, 1973. DOI: 10.1109/T-C.1973.223600.

[WDGS87] T. Williams, W. Daehn, M. Gruetzner, and C. Starke, “Aliasing Errors in Signature in Analysis Registers”, *IEEE Design & Test of Computers*, vol. 4, no. 2, pp. 39–45, Apr. 1987. DOI: 10.1109/MDT.1987.295105.

[WDGS88] T. Williams, W. Daehn, M. Gruetzner, and C. Starke, “Bounds and analysis of aliasing errors in linear feedback shift registers”, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 75–83, Jan. 1988. DOI: 10.1109/43.3132.

[WEF+85] J. A. Waicukauski, E. B. Eichelberger, D. O. Forlenza, E. Lindbloom, and T. Mccarthy, "Fault simulation for structured VLSI", *VLSI Systems Design*, vol. 6, no. 12, pp. 20–32, 1985.

[WT85] R. A. Walker and D. E. Thomas, "A Model of Design Representation and Synthesis", in *Proc. ACM/IEEE Design Automation Conference (DAC)*, 1985, pp. 453–459. DOI: 10.1109/DAC.1985.1585981.

[Wun10] H.-J. Wunderlich, Ed., *Models in Hardware Testing*, ser. Frontiers in Electronic Testing. Springer, 2010, vol. 43. DOI: 10.1007/978-90-481-3282-9.

[WWW+03] J. Wang, W. Wong, S. Wolday, B. Cronquist, J. McCollum, R. Katz, and I. Kleyner, "Single event upset and hardening in 0.15 μm antifuse-based field programmable gate array", *IEEE Trans. on Nuclear Science*, vol. 50, no. 6, pp. 2158–2166, 2003. DOI: 10.1109/TNS.2003.822090.

[WWW06] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*, ser. Systems on Silicon. Morgan Kaufmann, 2006, ISBN: 9780080474793.

[Xil14] Xilinx Inc. (Nov. 2014). Product Guide 036 - Soft Error Mitigation Controller v4.1, [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/sem/v4_1/pg036_sem.pdf.

[YHW98] V. Yarmolik, S. Hellebrand, and H.-J. Wunderlich, "Self-adjusting output data compression: An efficient BIST technique for RAMs", in *Proc. Design, Automation and Test in Europe (DATE)*, 1998, pp. 173–179. DOI: 10.1109/DATE.1998.655853.

[YKI+08] Y. Yanagawa, D. Kobayashi, H. Ikeda, H. Saito, and K. Hirose, "Scan-Architecture-Based Evaluation Technique of SET and SEU Soft-Error Rates at Each Flip-Flop in Logic VLSI Systems", *IEEE Trans. on Nuclear Science*, vol. 55, no. 4, pp. 1947–1952, Aug. 2008. DOI: 10.1109/TNS.2008.2000772.

[YNA09] H. Yu, M. Nicolaidis, and L. Anghel, "An effective approach to detect logic soft errors in digital circuits based on GRAAL", *Proc. International Symposium on Quality of Electronic Design (ISQED)*, pp. 236–240, 2009. DOI: 10.1109/ISQED.2009.4810300.

[ZMM+06] M. Zhang, S. Mitra, T. Mak, N. Seifert, N. Wang, Q. Shi, K. Kim, N. Shanbhag, and S. Patel, “Sequential Element Design With Built-In Soft Error Resilience”, *IEEE Trans. on VLSI Systems*, vol. 14, no. 12, pp. 1368–1378, 2006. DOI: 10.1109/TVLSI.2006.887832.

[ZMT+07] M. Zhang, T. Mak, J. Tschanz, K. Kim, N. Seifert, and D. Lu, “Design for Resilience to Soft Errors and Variations”, *Proc. IEEE Intl. On-Line Testing Symp. (IOLTS)*, pp. 23–28, 2007. DOI: 10.1109/IOLTS.2007.26.

Part IV

Appendices

Appendix A
Tables with Experimental Results

This appendix starts with details on the electronic design automation tools employed throughout this work (Section A.1) followed by the characteristics of the public and industrial benchmark circuits used during evaluation (Section A.2). The remaining sections exhibit additional experimental results in tabulated form for both parts of this work, the Fault Tolerance Infrastructure (Section A.3) as well as the Infrastructure Reuse for Offline Testing (Section A.4).

A.1. Electronic Design Automation Flow and Tools

The work at hand copes with fault tolerance and test by a new unified architecture. Dedicated solutions are used to solve specific challenges efficiently at different abstraction levels (see Section 2.1.1). Thus, the used *Electronic Design Automation* flow as depicted in Figure A.1 involves a wide range of tasks that reach from layout to circuit level. The tools used to conduct each task are listed in Table A.1.

Circuit Level (Chapters 4,5,7): Throughout this work, several *custom standard cells* are developed and implemented that constitute the foundation of the unified architecture. The *Parity-Pair Latch* to enable an area efficient register parity computation during non-concurrent detection. The *Bit-Flipping Latch* that offers an area efficient and fast concurrent online correction. And last but not least, the *Transmission-Gate Exclusive OR* which reduces the area overhead of the characteristic computation. Starting from the structural *schematic netlist*, the physical cell geometry is derived during *layout generation* and validated by comparison of the cell specification (schematic netlist) to its realization (layout netlist from *physical extraction*) during *layout-vs-schematic*. The electrical properties, the *timing behavior* as well as *power and energy consumption* are quantified for selected operating conditions during *analog simulation*.

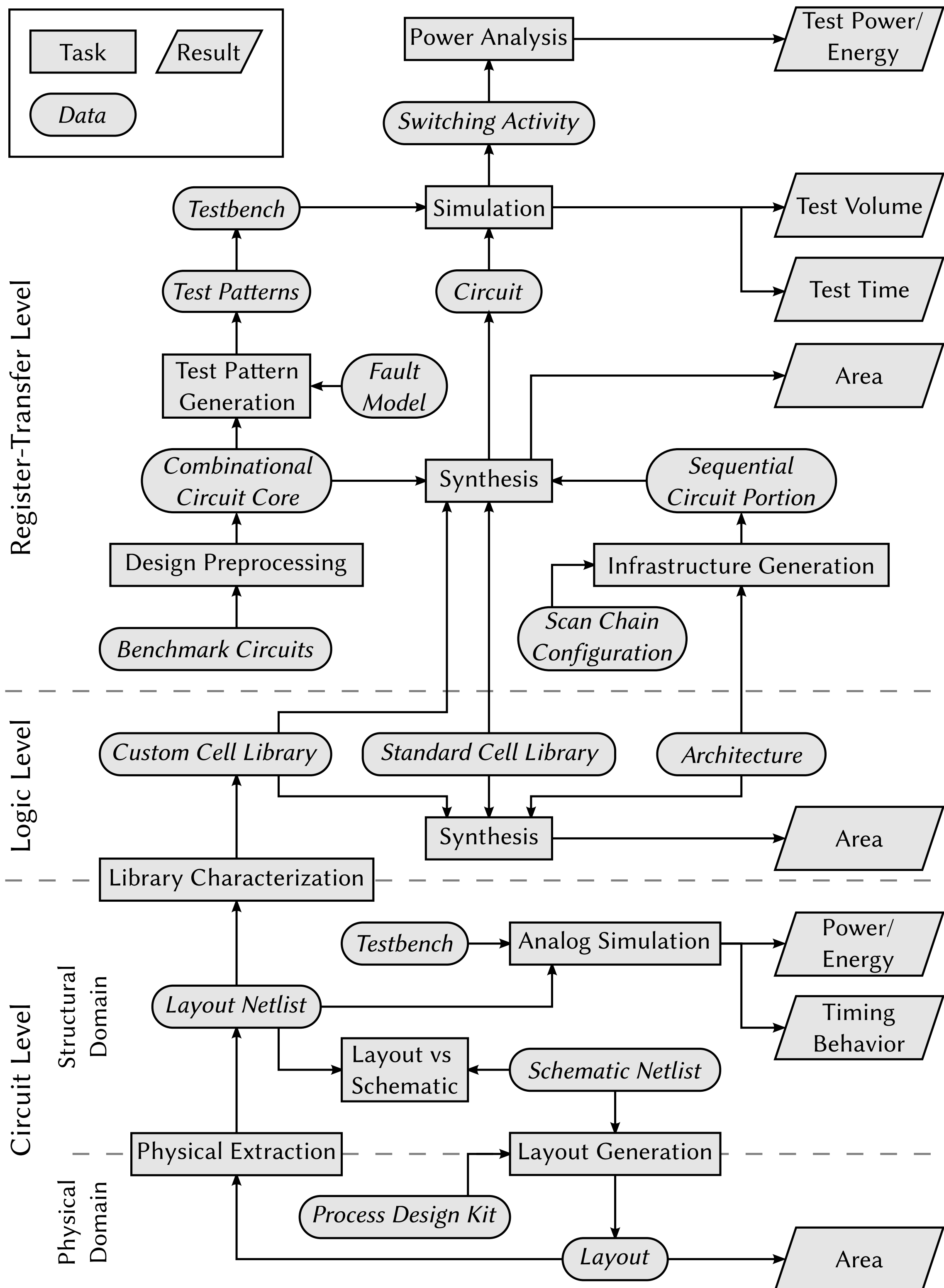

Figure A.1.: Experimental Setup with EDA Tool and Data Flow.

Table A.1.: Used Electronic Design Automation Tools and Versions.

Task	Vendor, Tool	Version
Power Analysis	Synopsys, PrimeTime	H-2012.12
Simulation	Mentor Graphics, ModelSim SE	6.4a
Test Pattern Generation		
- Scan	Mentor Graphics, Tessent FastScan	v9.5
- Bit-Flipping Scan	Inhouse, Computer Aided Test (CAT),	
	Inhouse, Sequential SAT ATPG (S^2ATPG)	
Infrastructure Generation	Inhouse, Soft Error (SERROR)	
Design Preprocessing	Inhouse, Design for X (DFX)	
Synthesis	Synopsys, Design Compiler	G-2012.06
Library Characterization	Cadence, Encounter Library Characterizer	v07.13
Analog Simulation	Synopsys, HSPICE	B-2008.09
Layout vs Schematic	Mentor Graphics, Calibre LVS	v2008.3
Physical Extraction	Mentor Graphics, Calibre PEX	v2008.3
Design Rule Check	Mentor Graphics, Calibre DRC	v2008.3
Layout Generation	Cadence, Virtuoso Layout Suite	IC6.1.3.1

Logic Level (Chapters 4-7,8): The cell properties are then automatically quantified for a larger range of operation conditions during *library characterization* which allows to abstract from cells to gates represented in a *custom cell library*. Together with a *standard cell library*, the different architectures and sequential payloads are synthesized for different register sizes in order to quantify their *area overhead*.

Register-Transfer Level (Chapters 8,9): *Circuits* outfitted with the architectures are *synthesized* by combining the *combinational circuit core* of benchmark circuits extracted during *design preprocessing* with a tailored *sequential circuit portion* obtained by architecture instantiation during *infrastructure generation*. *Test pattern generation* is conducted and the resulting *test patterns* are applied to the circuit via a *testbench* during *simulation* to quantify *test time* and *test volume*. The recorded *switching activity* finally allows to quantify *test power and energy* during *power analysis*.

Tools specific to this work are realized as modules of the internal EDA framework 'Design for X (DFX)' and implemented in Java ('Inhouse' in Table A.1). The circuit preparation from benchmarks to fully outfitted circuits is managed by the 'SERROR' module. The satisfiability-based test sequence generation from Chapter 9 targeting the Bit-Flipping Scan architecture is implemented within the 'S^2ATPG' module.

A.2. Benchmark Circuits

Throughout the experimental evaluation of the Unified Architecture in Part III sequential circuits are used that originate from three different sources. All benchmark circuits are modeled by a structural gate level description composed of gates with up to two inputs where gates with more inputs have been dissolved.

The used *public circuits* depicted in Table A.2 stem from the sets of digital sequential circuits presented at the International Symposium on Circuits and Systems in 1989

Table A.2.: Properties of the used Public Benchmark Circuits.

	Gates		Primary		
Name *(1)*	Combinational *(2)*	Sequential *(3)*	Inputs *(4)*	Outputs *(5)*	Depth *(6)*
s13207	8 668	669	31	121	65
s13207a	8 668	638	62	152	65
s15850	10 211	597	14	87	88
s15850a	10 211	534	77	150	88
s35932	16 353	1 728	35	320	30
s38417	23 537	1 636	28	106	49
s38584	21 462	1 452	12	278	60
s38584a	21 462	1 426	38	304	60
b14	10 735	245	32	54	67
b14a	7 459	245	32	54	61
b15	9 947	449	36	70	73
b15a	14 122	449	36	70	56
b17	35 549	1 415	37	97	104
b17a	42 879	1 415	37	97	56
b18	124 952	3 320	37	44	175
b18a	118 329	3 320	37	44	175
b19	251 692	6 642	24	71	180
b19a	238 942	6 642	24	71	180
b20	21 599	490	32	22	74
b20a	15 354	490	32	22	75
b21	22 055	490	32	22	75
b21a	15 460	490	32	22	71
b22	32 090	735	32	22	75
b22a	23 210	735	32	22	73

(ISCAS89, denoted by 's') [BBK89] and at the International Test Conference in 1999 (ITC99, denoted by 'b') [Dav99; CRS00]. Table A.2 denotes the properties of the selected public circuits which include the name (col. 1), the amount of combinational and sequential gates (col. 2 and 3), the number of primary in- and outputs (col. 4 and 5) as well as the circuit depth (col. 6).

Industrial designs were kindly provided by NXP (formerly Philips). Their properties in Table A.3 contain two additional columns that depict the number of scan chains in the original scan configuration and the length of the longest chain for reference.

Table A.3.: Properties of the used Industrial Benchmark Circuits.

	Gates		Primary			Scanchain Configuration	
Name *(1)*	Comb. *(2)*	Seq. *(3)*	Inputs *(4)*	Outputs *(5)*	Depth *(6)*	Chains *(7)*	Max. Length *(8)*
p35k	46 435	2 173	739	56	72	23	100
p45k	43 190	2 331	1 408	219	61	97	333
p77k	72 370	3 386	101	14	570	13	304
p78k	74 243	2 977	171	507	47	65	64
p81k	108 991	3 877	152	75	56	8	485
p89k	88 726	4 301	331	256	111	18	963
p100k	96 685	5 735	167	94	106	18	792
p141k	172 686	10 501	789	1	80	24	486
p239k	259 241	18 382	310	113	185	40	541
p259k	334 524	18 398	315	97	188	40	541
p267k	271 538	16 528	804	93	75	45	494
p269k	272 630	16 528	805	93	75	45	494
p279k	287 939	17 524	554	311	151	55	409
p286k	364 347	17 713	638	122	155	55	416
p295k	291 022	18 465	43	56	115	11	1 852
p330k	355 642	16 775	1 235	693	72	64	317
p378k	371 215	14 885	847	2 535	47	325	64
p388k	489 271	23 789	1 216	276	226	50	525
p418k	439 198	28 616	1 814	1 193	207	64	830
p469k	75 572	332	303	71	175	1	332
p483k	515 717	32 307	957	303	110	71	900
p500k	495 544	29 312	1 456	1 528	171	76	446
p533k	662 730	32 409	964	201	114	71	900
p874k	717 302	41 803	1 108	454	240	59	780

A.3. Results - Fault Tolerance Infrastructure

Tabulated results and absolute area values for the four chapters of Part II are provided in the following.

A.3.1. Non-Concurrent Detection and Localization of Single Event Upsets

Chapter 4 focuses on protecting modules equipped with clock gating against soft errors during phases where the clock is disabled to increase power efficiency.

Single Event Upset Detection at Gate Level (Chapter 4.2) Table A.4 reports the area overhead associated with the register parity computation across different register sizes (col. 1) for a reference standard cell implementation (col. 3-4) and the presented Parity-Pair Latch (col. 5-6) over the original register (col. 2).

Single Event Upset Localization at Module Level (Chapter 4.3) Table A.5 shows the area overhead as a function of register size (col. 1) and register count (col. 2 divided by col. 1), if the modulo-2 address characteristic is used to implement a module wide localization on top of multiple registers equipped with above mentioned reference and parity-pair latch based parity computation.

Table A.4.: Area Overhead - Parity Computation for a Single Register - Reference Implementation (OCL) and Parity-Pair Latch (PPL) (visualized in Figure 4.12).

		Detection (Fig. 4.1-b and 4.1-c)			
Register	Unprotected (Fig. 4.1-a)	Reference (OCL, Fig. 4.2)		Parity-Pair Latch (PPL, Fig. 4.4)	
Size [*bit*] *(1)*	Area [μm^2] *(2)*	Area [μm^2] *(3)*	Overhead [+%] *(4)*	Area [μm^2] *(5)*	Overhead [+%] *(6)*
8	21.28	38.84	+82.52%	22.88	+7.52%
16	42.56	79.27	+86.25%	47.35	+11.25%
32	85.12	160.13	+88.12%	96.29	+13.12%
64	170.24	321.86	+89.06%	194.18	+14.06%
128	340.48	645.32	+89.53%	389.96	+14.53%

Table A.5.: Area Overhead - SEU Localization across Multiple Registers - Reference Implementation (OCL) and Parity-Pair Latch (PPL) (visualized in Figure 4.13).

Size		Unprotected	Detection & Localization (Fig. 4.1-c)			
		(Fig. 4.1-a)	Reference (OCL)		Parity Pair Latch (PPL)	
Register [*bit*] *(1)*	Total [*bit*] *(2)*	Area [μm^2] *(3)*	Area [μm^2] *(4)*	Overhead [+%] *(5)*	Area [μm^2] *(6)*	Overhead [+%] *(7)*
8	256	680.96	1355.80	+99.10%	840.29	+23.40%
	512	1361.92	2702.56	+98.44%	1675.53	+23.03%
	1024	2723.84	5393.95	+98.03%	3344.68	+22.79%
	2048	5447.68	10775.66	+97.80%	6682.72	+22.67%
	4096	10895.36	21537.49	+97.68%	13357.99	+22.60%
16	256	680.96	1328.14	+95.04%	813.43	+19.45%
	512	1361.92	2649.63	+94.55%	1623.40	+19.20%
	1024	2723.84	5290.21	+94.22%	3242.27	+19.03%
	2048	5447.68	10569.24	+94.01%	6477.10	+18.90%
	4096	10895.36	21126.25	+93.90%	12947.55	+18.84%
32	256	680.96	1313.24	+92.85%	799.33	+17.38%
	512	1361.92	2621.96	+92.52%	1596.53	+17.23%
	1024	2723.84	5237.27	+92.28%	3189.61	+17.10%
	2048	5447.68	10466.04	+92.12%	6374.69	+17.02%
	4096	10895.36	20919.84	+92.01%	12741.93	+16.95%
64	256	680.96	1305.53	+91.72%	792.41	+16.37%
	512	1361.92	2607.07	+91.43%	1582.43	+16.19%
	1024	2723.84	5209.61	+91.26%	3162.74	+16.11%
	2048	5447.68	10412.57	+91.14%	6322.02	+16.05%
	4096	10895.36	20816.63	+91.06%	12639.52	+16.01%
128	256	680.96	1300.21	+90.94%	787.89	+15.70%
	512	1361.92	2599.35	+90.86%	1575.52	+15.68%
	1024	2723.84	5194.71	+90.71%	3148.64	+15.60%
	2048	5447.68	10384.91	+90.63%	6295.16	+15.56%
	4096	10895.36	20763.16	+90.57%	12586.85	+15.52%

A.3.2. Concurrent Online Correction of Single Event Upsets

In Chapter 5, the clocked phase of a module is protected by deriving register specific error conditions through directly applying the modulo-2 address characteristic.

Single Error Detection (SED) (Chapter 5.2) Table A.6 depicts the area overhead associated with the derivation and the protected storage of the register specific error condition in order to provide single error detection (col. 5-6) in comparison to bitwise duplication with comparison (col. 3-4).

Single Error Correction (SEC) (Chapter 5.3) Table A.7 list the area overhead if the architecture is extended to a self-contained rapid correction by replacing the register latches with the developed Bit-Flipping Latches (col. 7-8). In addition to the unprotected area (col. 2), the area and overhead of bitwise triple modular redundancy (col. 3-4) and bitwise fault tolerance (col. 5-6) are provided for reference.

A.3.3. Fault Tolerance in Presence of Multiple Bit Upsets

Chapter 6 investigates the behavior with respect to multiple errors and extends the detection capabilities to double errors by utilizing the extended characteristic.

Online Architecture for Double Errors (Chapter 6.2) Table A.8 depicts the area overhead for sole double error detection (col. 3-4) and its combination with single error correction (col. 5-6).

Table A.6.: Area Overhead - Single Error Detection (SED) (visualized in Figure 5.5).

Register	Unprotected (Fig. 5.1-a)	Duplication with Comparison (DWC)		Detecting (Fig. 5.1-b)	
Size [bit] *(1)*	Area [μm^2] *(2)*	Area [μm^2] *(3)*	Overhead [+%] *(4)*	Area [μm^2] *(5)*	Overhead [+%] *(6)*
7	18.62	53.73	+188.56%	51.60	+177.12%
15	39.90	115.44	+189.32%	94.16	+135.99%
31	82.46	239.13	+190.00%	177.95	+115.80%
63	167.58	485.72	+189.84%	342.61	+104.45%
127	337.82	979.68	+190.00%	668.72	+97.95%
255	678.30	1966.80	+189.96%	1318.03	+94.31%

Table A.7.: Area Overhead - Single Error Correction (SEC) (visualized in Figure 5.9).

Register	Unprotected (Fig. 5.1-a)	Triple Modular Redundancy (TMR)		Fault Tolerance (FT)		Correcting (Fig. 5.1-c)	
Size [*bit*] *(1)*	Area [μm^2] *(2)*	Area [μm^2] *(3)*	Overhead [+%] *(4)*	Area [μm^2] *(5)*	Overhead [+%] *(6)*	Area [μm^2] *(7)*	Overhead [+%] *(8)*
7	18.62	80.07	+330.02%	79.80	+328.57%	81.93	+340.01%
15	39.90	171.57	+330.00%	171.30	+329.32%	148.69	+272.66%
31	82.46	354.58	+330.00%	354.84	+330.32%	287.55	+248.71%
63	167.58	720.59	+330.00%	721.92	+330.79%	551.15	+228.89%
127	337.82	1452.63	+330.00%	1457.41	+331.42%	1060.01	+213.78%
255	678.30	2916.69	+330.00%	2926.80	+331.49%	2114.17	+211.69%

A.3.4. Area Efficient Characteristic Computation

Chapter 7 first analyzes the area overhead of the single error correction architecture and identifies the characteristic computation as a major area contributor. The developed area efficient Transmission-Gate Exclusive XOR is shown to further reduce the overhead and to be beneficial for all previously presented architectures.

Table A.8.: Area Overhead - Single and Double Error Detection (DED), Single Error Correction Double Error Detection (SECDED) - Single Register (visualized in Figure 6.3).

Register	Unprotected (Fig. 5.1-a)	Double Detecting (DED) (Fig. 5.1-b & Fig. 6.1)		Single Correcting, Double Detecting (SECDED) (Fig. 5.1-c & Fig. 6.1)	
Size [*bit*] *(1)*	Area [μm^2] *(2)*	Area [μm^2] *(3)*	Overhead [+%] *(4)*	Area [μm^2] *(5)*	Overhead [+%] *(6)*
7	18.62	59.32	+218.58%	89.64	+381.42%
15	39.90	103.47	+159.32%	158.00	+295.99%
31	82.46	188.86	+129.03%	298.45	+261.93%
63	167.58	355.11	+111.90%	563.65	+236.35%
127	337.82	682.82	+102.13%	1074.11	+217.95%
255	678.30	1333.72	+96.63%	2129.86	+214.00%

Detailed Analysis of the Correction Area Overhead (Chapter 7.1) Table A.9 shows the detailed area overhead breakdown of single error correction. The total area (col. 9) is partitioned and attributed to the Bit-Flipping Latches (col. 3), the characteristic tree (col. 4), the checksum register and syndrome derivation (col. 5), its parity protection (col. 6), the syndrome decoder (col. 7), and the top level (col. 8).

Table A.9.: Detailed Area Overhead Analysis of Single Error Correction (SEC) Components (see Table A.7) (visualized in Figure 7.1).

Register	Unprotected	Correcting (Fig. 5.1-c)						
	(DLH_X1)	BFL	Char	S/fail S	fail P	Decoder	Top	Sum
Size	Area	Area	Area	Area	Area	Area	Area	Area
$[bit]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$
(1)	*(2)*	*(3)*	*(4)*	*(5)*	*(6)*	*(7)*	*(8)*	*(9)*
7	18.62	22.34	12.77	12.51	6.65	19.15	8.51	81.93
15	39.90	47.88	35.11	16.22	8.25	32.72	8.51	148.69
31	82.46	98.95	82.99	20.75	9.84	65.97	9.05	287.55
63	167.58	201.10	181.94	25.27	11.44	121.03	10.37	551.15
127	337.82	405.38	383.04	29.52	13.04	217.06	11.97	1060.01
255	678.30	813.96	788.42	33.52	14.63	448.47	15.17	2114.17

Area Efficient Exclusive OR Trees (Chapter 7.2) Table A.10 shows the area breakdown after applying the Transmission-Gate XOR to the characteristic computation (col. 4), the syndrome comparator (col. 5) and the characteristic protection (col. 6).

Table A.10.: Detailed Area Overhead Analysis of the Area Efficient Single Error Correction (SEC TG) utilizing the Transmission-Gate XOR (vis. in Fig. 7.7).

Register	Unprotected	Area Efficient Correcting (Fig. 5.1-c & Fig. 7.3)						
	(DLH_X1)	BFL	Char	S/fail S	fail P	Decoder	Top	Sum
Size	Area	Area	Area	Area	Area	Area	Area	Area
$[bit]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$	$[\mu m^2]$
(1)	*(2)*	*(3)*	*(4)*	*(5)*	*(6)*	*(7)*	*(8)*	*(9)*
7	18.62	22.34	6.38	10.11	5.06	19.15	9.05	72.09
15	39.90	47.88	17.56	13.03	5.85	32.72	9.04	126.08
31	82.46	98.95	41.50	16.76	6.65	65.97	9.57	239.40
63	167.58	201.10	90.97	20.48	8.25	121.03	10.90	452.73
127	337.82	405.38	191.52	23.94	8.25	217.06	12.50	858.65
255	678.30	813.96	394.21	27.13	9.05	448.47	15.70	1708.52

Table A.11 depicts the area overhead for the area efficient detection of single errors (col. 3-4) and double errors (col. 5-6).

Table A.11.: Area Overhead - Area Efficient Error Detection (SED TG, DED TG) (visualized in Figure 7.8).

		Area Efficient Detecting (Fig. 5.1-b & Fig. 7.3)			
		Single Detecting (SED TG)		Double Detecting (DED TG) (& Fig. 6.1)	
Register	Unprotected				
Size [*bit*] *(1)*	Area [μm^2] *(2)*	Area [μm^2] *(3)*	Overhead [+%] *(4)*	Area [μm^2] *(5)*	Overhead [+%] *(6)*
7	18.62	41.23	+121.43%	47.35	+154.30%
15	39.90	71.02	+77.99%	77.94	+95.34%
31	82.46	129.28	+56.78%	136.99	+66.13%
63	167.58	243.66	+45.40%	252.17	+50.48%
127	337.82	466.83	+38.19%	476.14	+40.94%
255	678.30	911.85	+34.43%	921.96	+35.92%

Table A.12 lists the area overhead for the area efficient correction combined with single error detection (col. 3-4) and double error detection (col. 5-6).

Table A.12.: Area Overhead - Area Efficient Error Correction (SEC TG, SECDED TG) (visualized in Figure 7.8).

		Area Efficient Correcting (Fig. 5.1-c & Fig. 7.3)			
		Single Correcting (SEC TG)		Single Correcting, Double Detecting (SECDED TG) (& Fig. 6.1)	
Register	Unprotected				
Size [*bit*] *(1)*	Area [μm^2] *(2)*	Area [μm^2] *(3)*	Overhead [+%] *(4)*	Area [μm^2] *(5)*	Overhead [+%] *(6)*
7	18.62	72.09	+287.16%	77.67	+317.13%
15	39.90	126.08	+215.99%	132.47	+232.01%
31	82.46	239.40	+190.32%	246.58	+199.03%
63	167.58	452.73	+170.16%	460.71	+174.92%
127	337.82	858.65	+154.17%	867.43	+156.77%
255	678.30	1708.52	+151.88%	1718.09	+153.29%

A.4. Results - Infrastructure Reuse for Offline Testing

Part III focuses on offline testing, achieves full testability and targets two domains to increase test efficiency. Test access is provided by reusing the fault tolerance infrastructure from the last part. To capitalize the unique properties of the provided test access, customized test sequences are generated. Thus, all secondary test metrics from Section 1.2.2 are evaluated.

A.4.1. Test Access through Infrastructure Reuse

Chapter 8 extends the previously developed concurrent fault tolerance architecture. The resulting unified 'Bit-Flipping Scan' architecture doubles as a test access mechanism during offline test and requires only small amendments.

Unified Architecture (Chapter 8.1) Table A.13 discloses the area overhead if single registers of different sizes (col. 1) are equipped with scan design (col. 3-4), the combination of scan design with bitwise fault tolerance (col. 5-6), Bit-Flipping Scan (col. 7-8) as well as Area Efficient Bit-Flipping Scan (col. 9-10).

The area overhead associated with the application to complete circuits is provided in Table A.14 for public circuits and in Table A.15 for industrial circuits.

Table A.13.: Unified Architecture - Area Overhead for a Single Register equipped with Scan, Fault Tolerance & Scan, Bit Flipping Scan and Area Efficient Bit Flipping Scan (visualized in Figure 10.2).

Reg.	Original	Scan Design		Fault Tolerance & Scan Design		Bit-Flipping Scan		Area Efficient Bit-Flipping Scan	
Size [*bit*] *(1)*	(DFF) [μm^2] *(2)*	(SDFFR) [μm^2] *(3)*	OH [+%] *(4)*	(SDFFR) [μm^2] *(5)*	OH [+%] *(6)*	(BFFF) [μm^2] *(7)*	OH [+%] *(8)*	(BFFF) [μm^2] *(9)*	OH [+%] *(10)*
7	31.65	47.61	+50.43%	97.09	+206.76%	115.71	+265.59%	105.34	+232.83%
15	67.83	100.81	+48.62%	206.15	+203.92%	201.10	+196.48%	177.95	+162.35%
31	140.18	209.87	+49.71%	427.46	+204.94%	374.00	+166.80%	326.12	+132.64%
63	284.89	425.07	+49.20%	863.97	+203.26%	700.64	+145.93%	601.69	+111.20%
127	574.29	853.33	+48.59%	1743.10	+203.52%	1331.86	+131.91%	1129.97	+96.76%
255	1153.11	1720.75	+49.23%	3502.16	+203.71%	2590.84	+124.68%	2184.66	+89.46%

A.4.2. Test Sequence Generation

Chapter 9 provides heuristically generated tailored test sequences that are able to fully exploit the test access capabilities provided by the 'Bit-Flipping Scan' architecture.

Bit-Flipping Scan Test Sequence Generation (Chapter 9.3) Table A.16 and Table A.17 list the test application time (col. 2-11) for the public respectively the industrial circuits. For the reference test set, the number of patterns (col. 2) as well as the amount of test cycles needed for application through classical scan design (col. 5) are provided along with the number of scan (col. 3) and capture cycles (col. 4). For the Bit-Flipping Scan test sequence, the pattern count (col. 6) is depicted along with the test cycles used during scan (col. 7), flip (col. 8) and capture (col. 9) operations as well as the total number of used cycles (col. 10). Finally, the test time speedup is calculated for comparison (col. 11).

The test data volume, the amount of bits exchanged externally with the test access mechanism, for both schemes is contained in column 12 respectively 13 of Table A.16 and Table A.17. The ratio between both volumes (col. 14) determines the test volume reduction achieved by the Bit-Flipping Scan architecture.

Table A.18 and Table A.19 provide insight to the peak (col. 2-4) and average power (col. 5-7) consumption of the used public and industrial circuits during test application.

The remaining columns of Table A.18 and Table A.19 finally denote the test energy consumption (col. 8-10), the product of test time and average test power, for the public and industrial circuits.

Table A.14.: Unified Architecture - Area Overhead for Public Circuits (partially visualized in Fig. 10.3).

	Original	Scan Design		Fault Tolerance & Scan Design		Bit-Flipping Scan		**Area**	Area Efficient Bit-Flipping Scan		**Area**
Circuit *name* *(1)*	(DFF) [μm^2] *(2)*	(SDFFR) [μm^2] *(3)*	OH [+%] *(4)*	(SDFFR) [μm^2] *(5)*	OH [+%] *(6)*	(BFFF) [μm^2] *(7)*	OH [+%] *(8)*	**ΔOH** **[+%]** *(8)-(6)*	(BFFF) [μm^2] *(10)*	OH [+%] *(11)*	**ΔOH** **[+%]** *(11)-(6)*
s13207	9 115	10 554	15.8	15 332	68.2	13 135	44.1	**-24.1**	12 172	33.5	**-34.7**
s13207a	8 975	10 348	15.3	14 895	66.0	12 777	42.4	**-23.6**	11 815	31.6	**-34.3**
s15850	9 898	11 183	13.0	15 447	56.0	13 467	36.0	**-20.0**	12 697	28.3	**-27.8**
s15850a	9 614	10 763	11.9	14 574	51.6	12 824	33.4	**-18.2**	12 054	25.4	**-26.2**
s35932	21 216	24 933	17.5	37 250	75.6	31 480	48.4	**-27.2**	28 977	36.6	**-39.0**
s38417	24 066	27 585	14.6	39 242	63.1	33 765	40.3	**-22.8**	31 455	30.7	**-32.4**
s38584	24 328	27 451	12.8	37 826	55.5	32 960	35.5	**-20.0**	30 842	26.8	**-28.7**
s38584a	24 210	27 278	12.7	37 436	54.6	32 693	35.0	**-19.6**	30 575	26.3	**-28.3**
b14	9 911	10 437	5.3	12 180	22.9	11 385	14.9	**-8.0**	11 192	12.9	**-10.0**
b14a	7 252	7 779	7.3	9 521	31.3	8 727	20.3	**-11.0**	8 534	17.7	**-13.6**
b15	10 451	11 418	9.2	14 625	39.9	13 185	26.2	**-13.8**	12 608	20.6	**-19.3**
b15a	13 649	14 615	7.1	17 820	30.6	16 396	20.1	**-10.4**	15 818	15.9	**-14.7**
b17	36 049	39 091	8.4	49 172	36.4	44 587	23.7	**-12.7**	42 469	17.8	**-18.6**
b17a	41 682	44 724	7.3	54 802	31.5	50 213	20.5	**-11.0**	48 095	15.4	**-16.1**
b18	116 499	123 628	6.1	147 287	26.4	136 697	17.3	**-9.1**	131 691	13.0	**-13.4**
b18a	111 013	118 142	6.4	141 799	27.7	131 196	18.2	**-9.6**	126 190	13.7	**-14.1**
b19	234 294	248 556	6.1	295 892	26.3	274 661	17.2	**-9.1**	264 650	13.0	**-13.3**
b19a	223 854	238 115	6.4	285 443	27.5	264 162	18.0	**-9.5**	254 151	13.5	**-14.0**
b20	19 819	20 872	5.3	24 368	22.9	22 758	14.8	**-8.1**	22 180	11.9	**-11.0**
b20a	14 704	15 757	7.2	19 253	30.9	17 627	19.9	**-11.1**	17 049	15.9	**-15.0**
b21	20 266	21 319	5.2	24 815	22.4	23 205	14.5	**-7.9**	22 627	11.7	**-10.8**
b21a	14 849	15 903	7.1	19 398	30.6	17 772	19.7	**-10.9**	17 194	15.8	**-14.8**
b22	29 535	31 115	5.3	36 366	23.1	33 925	14.9	**-8.3**	32 962	11.6	**-11.5**
b22a	22 214	23 794	7.1	29 044	30.8	26 590	19.7	**-11.1**	25 627	15.4	**-15.4**

Table A.15.: Unified Architecture - Area Overhead for Industrial Circuits (partially visualized in Fig. 10.3).

	Original	Scan Design		Fault Tolerance & Scan Design		Bit-Flipping Scan		**Area**	Area Efficient Bit-Flipping Scan		**Area**
Circuit *name* *(1)*	(DFF) [μm^2] *(2)*	(SDFFR) [μm^2] *(3)*	OH [+%] *(4)*	(SDFFR) [μm^2] *(5)*	OH [+%] *(6)*	(BFFF) [μm^2] *(7)*	OH [+%] *(8)*	**ΔOH** **[+%]** *(8)-(6)*	(BFFF) [μm^2] *(10)*	OH [+%] *(11)*	**ΔOH** **[+%]** *(11)-(6)*
p35k	48 819	53 494	9.6	68 980	41.3	61 893	26.8	**-14.5**	58 620	20.1	**-21.2**
p45k	47 605	52 620	10.5	69 229	45.4	61 624	29.4	**-16.0**	58 159	22.2	**-23.2**
p77k	78 481	85 765	9.3	109 901	40.0	98 816	25.9	**-14.1**	93 811	19.5	**-20.5**
p78k	81 305	87 708	7.9	108 917	34.0	99 191	22.0	**-12.0**	94 763	16.6	**-17.4**
p81k	116 490	124 831	7.2	152 462	30.9	139 783	20.0	**-10.9**	134 008	15.0	**-15.8**
p89k	98 019	107 271	9.4	137 924	40.7	123 809	26.3	**-14.4**	117 456	19.8	**-20.9**
p100k	109 949	122 286	11.2	163 164	48.4	144 385	31.3	**-17.1**	135 721	23.4	**-25.0**
p141k	196 445	219 033	11.5	293 864	49.6	259 432	32.1	**-17.5**	243 645	24.0	**-25.6**
p239k	312 478	352 019	12.7	483 004	54.6	422 712	35.3	**-19.3**	394 989	26.4	**-28.2**
p259k	382 826	422 400	10.3	553 499	44.6	493 141	28.8	**-15.8**	465 418	21.6	**-23.0**
p267k	293 051	328 604	12.1	446 409	52.3	392 201	33.8	**-18.5**	367 173	25.3	**-27.0**
p269k	294 209	329 762	12.1	447 567	52.1	393 359	33.7	**-18.4**	368 331	25.2	**-26.9**
p279k	326 800	364 495	11.5	489 402	49.8	431 890	32.2	**-17.6**	405 514	24.1	**-25.7**
p286k	399 257	437 358	9.5	563 607	41.2	505 505	26.6	**-14.5**	478 744	19.9	**-21.2**
p295k	341 412	381 131	11.6	512 701	50.2	452 172	32.4	**-17.7**	424 257	24.3	**-25.9**
p330k	379 204	415 289	9.5	534 819	41.0	479 826	26.5	**-14.5**	454 414	19.8	**-21.2**
p378k	406 522	438 541	7.9	544 636	34.0	495 805	22.0	**-12.0**	473 280	16.4	**-17.5**
p388k	539 441	590 613	9.5	760 186	40.9	682 126	26.4	**-14.5**	646 125	19.8	**-21.1**
p418k	498 894	560 449	12.3	764 424	53.2	670 521	34.4	**-18.8**	627 204	25.7	**-27.5**
p469k	69 937	70 652	1.0	73 015	4.4	71 943	2.9	**-1.5**	71 558	2.3	**-2.1**
p483k	588 359	657 853	11.8	888 051	50.9	782 111	32.9	**-18.0**	733 211	24.6	**-26.3**
p500k	567 181	630 232	11.1	839 105	47.9	742 950	31.0	**-16.9**	698 671	23.2	**-24.8**
p533k	727 893	797 607	9.6	1 028 533	41.3	922 260	26.7	**-14.6**	873 168	20.0	**-21.3**
p874k	784 613	874 534	11.5	1 172 409	49.4	1 035 307	31.9	**-17.5**	971 968	23.9	**-25.6**

Table A.16.: Test Time (TAT) and Test Volume (TDV) for Public Circuits (partially visualized in Fig. 10.4 and 10.5).

	Test Application Time										**Test Data Volume**		
	FT + Scan Design				Bit-Flipping Scan					**TAT**			**TDV**
Circuit *name* *(1)*	Pat. [*cnt*] *(2)*	Capt. [*cyc*] *(3)*	Scan [*cyc*] *(4)*	Sum [*cyc*] *(5)*	Pat. [*cnt*] *(6)*	Capt. [*cyc*] *(7)*	Flip [*cyc*] *(8)*	Scan [*cyc*] *(9)*	Sum [*cyc*] *(10)*	**Speedup** **[*ratio*]** *(11)*	FTScan [*bit*] *(12)*	BFScan [*bit*] *(13)*	**Reduction** **[*%*]** *(14)*
s13207	311	311	39 624	39 935	487	487	638	4 366	5 491	**7.27**	463 390	113 577	**75.49**
s13207a	293	293	37 338	37 631	479	479	657	4 419	5 555	**6.77**	436 570	136 930	**68.64**
s15850	171	171	21 844	22 015	529	529	644	4 433	5 606	**3.92**	221 445	90 674	**59.06**
s15850a	185	185	23 622	23 807	515	515	758	5 011	6 284	**3.78**	239 575	150 605	**37.14**
s35932	62	62	8 001	8 063	112	112	113	798	1 023	**7.88**	236 282	61 675	**73.90**
s38417	168	168	21 463	21 631	588	588	596	4 177	5 361	**4.03**	572 208	185 651	**67.56**
s38584	197	197	25 146	25 343	544	544	558	3 909	5 011	**5.05**	629 218	248 007	**60.59**
s38584a	200	200	25 527	25 727	504	504	534	3 731	4 769	**5.39**	638 800	254 905	**60.10**
b14	771	771	98 044	98 815	1 016	1 016	1 481	9 460	11 957	**8.26**	444 096	117 187	**73.62**
b14a	600	600	76 327	76 927	876	876	1 862	11 307	14 045	**5.47**	345 600	105 027	**69.62**
b15	521	521	66 294	66 815	835	835	1 489	9 935	12 259	**5.45**	523 084	144 093	**72.46**
b15a	1 242	1 242	157 861	159 103	1 551	1 551	4 864	31 855	38 270	**4.15**	1 246 968	309 885	**75.15**
b17	1 108	1 108	140 843	141 951	2 196	2 196	3 369	23 400	28 965	**4.90**	3 284 112	672 204	**79.54**
b17a	1 287	1 287	163 576	164 863	3 375	3 375	5 960	41 279	50 614	**3.25**	3 814 668	1 043 850	**72.64**
b18	1 216	1 216	154 559	155 775	5 239	5 239	6 152	43 024	54 415	**2.86**	8 172 736	2 363 962	**71.08**
b18a	1 195	1 195	151 892	153 087	4 970	4 970	5 920	41 395	52 285	**2.92**	8 031 595	2 242 343	**72.09**
b19	1 417	1 417	180 086	181 503	6 734	6 734	6 918	48 433	62 085	**2.92**	18 958 043	5 610 535	**70.41**
b19a	1 417	1 417	180 086	181 503	6 374	6 374	6 544	45 815	58 733	**3.09**	18 958 043	5 310 441	**71.99**
b20	861	861	109 474	110 335	1 488	1 488	2 050	13 953	17 491	**6.30**	890 274	169 801	**80.93**
b20a	583	583	74 168	74 751	1 027	1 027	1 393	9 435	11 855	**6.30**	602 822	115 461	**80.85**
b21	868	868	110 363	111 231	1 559	1 559	2 158	14 623	18 340	**6.06**	897 512	177 564	**80.22**
b21a	604	604	76 835	77 439	1 098	1 098	1 501	10 192	12 791	**6.05**	624 536	124 363	**80.09**
b22	743	743	94 488	95 231	1 549	1 549	1 955	13 527	17 031	**5.59**	1 132 332	218 010	**80.75**
b22a	586	586	74 549	75 135	1 185	1 185	1 480	10 209	12 874	**5.83**	893 064	165 159	**81.51**

Table A.17.: Test Time (TAT) and Test Volume (TDV) for Industrial Circuits (partially visualized in Fig. 10.4 and 10.5).

	Test Application Time										**Test Data Volume**		
	FT + Scan Design				Bit-Flipping Scan					**TAT**			**TDV**
Circuit *name* *(1)*	Pat. [*cnt*] *(2)*	Capt. [*cyc*] *(3)*	Scan [*cyc*] *(4)*	Sum [*cyc*] *(5)*	Pat. [*cnt*] *(6)*	Capt. [*cyc*] *(7)*	Flip [*cyc*] *(8)*	Scan [*cyc*] *(9)*	Sum [*cyc*] *(10)*	**Speedup [*ratio*]** *(11)*	FTScan [*bit*] *(12)*	BFScan [*bit*] *(13)*	**Reduction [%]** *(14)*
p35k	1 438	1 438	182 753	184 191	1 964	1 964	2 194	15 349	19 507	**9.44**	7 392 758	2 048 432	**72.30**
p45k	2 101	2 101	266 954	269 055	2 463	2 463	2 558	17 903	22 924	**11.73**	13 213 189	4 655 608	**64.77**
p77k	541	541	68 834	69 375	1 551	1 551	1 577	11 043	14 171	**4.89**	3 725 867	762 934	**79.53**
p78k	85	85	10 922	11 007	111	111	110	777	998	**11.02**	563 720	112 332	**80.08**
p81k	383	383	48 768	49 151	1 733	1 733	1 792	12 549	16 074	**3.05**	3 056 723	1 144 175	**62.57**
p89k	804	804	102 235	103 039	3 321	3 321	3 440	24 084	30 845	**3.34**	7 387 956	3 525 669	**52.28**
p100k	2 055	2 055	261 112	263 167	2 757	2 757	3 182	22 280	28 219	**9.32**	24 107 205	2 459 545	**89.80**
p141k	684	684	86 995	87 679	1 255	1 255	1 264	8 855	11 374	**7.70**	14 905 728	2 448 999	**83.58**
p239k	590	590	75 057	75 647	1 659	1 659	1 704	11 935	15 298	**4.94**	21 940 330	4 059 772	**81.50**
p259k	719	719	91 440	92 159	1 764	1 764	1 780	12 467	16 011	**5.75**	26 752 552	4 302 357	**83.92**
p267k	906	906	115 189	116 095	2 033	2 033	2 150	15 057	19 240	**6.03**	30 761 418	5 526 438	**82.04**
p269k	925	925	117 602	118 527	2 031	2 031	2 144	15 015	19 190	**6.17**	31 407 450	5 523 488	**82.42**
p279k	874	874	111 125	111 999	2 209	2 209	2 268	15 883	20 360	**5.50**	31 387 962	6 166 916	**80.36**
p286k	1 172	1 172	148 971	150 143	3 666	3 666	3 731	26 123	33 520	**4.47**	42 409 992	9 953 294	**76.54**
p295k	1 852	1 852	235 331	237 183	5 373	5 373	5 804	40 635	51 812	**4.57**	68 577 708	11 421 548	**83.35**
p330k	2 291	2 291	291 084	293 375	3 650	3 650	3 673	25 718	33 041	**8.87**	81 280 098	13 807 024	**83.02**
p378k	83	83	10 668	10 751	156	156	155	1 092	1 403	**7.66**	2 751 616	784 680	**71.49**
p388k	609	609	77 470	78 079	1 675	1 675	1 690	11 837	15 202	**5.13**	29 883 630	6 904 909	**76.90**
p418k	963	963	122 428	123 391	3 345	3 345	3 454	24 185	30 984	**3.98**	58 010 157	20 602 670	**64.49**
p469k	403	403	51 308	51 711	451	451	559	3 798	4 808	**10.75**	418 314	187 539	**55.17**
p483k	385	385	49 022	49 407	2 202	2 202	2 220	15 547	19 969	**2.47**	25 361 490	10 624 855	**58.11**
p500k	2 376	2 376	301 879	304 255	5 885	5 885	6 821	47 754	60 460	**5.03**	146 380 608	35 863 584	**75.50**
p533k	575	575	73 152	73 727	2 296	2 296	2 367	16 576	21 239	**3.47**	37 940 225	10 878 569	**71.33**
p874k	1 298	1 298	164 973	166 271	3 228	3 228	3 342	23 401	29 971	**5.54**	110 548 064	19 740 183	**82.15**

Table A.18.: Peak and Average Test Power (TP) and Test Energy (TE) for Public Circuits (partially visualized in Fig. 10.7, Fig. 10.8 and Fig. 10.9).

	Test Power						**Test Energy**		
	Peak			**Average**					
Circuit *name* *(1)*	FTScan $[W]$ *(2)*	BFScan $[W]$ *(3)*	**Red. [%]** *(4)*	FTScan $[W]$ *(5)*	BFScan $[W]$ *(6)*	**Red. [%]** *(7)*	FTScan $[nJ]$ *(8)*	BFScan $[nJ]$ *(9)*	**Red. [%]** *(10)*
s13207	$8.30{\cdot}10^{-3}$	$1.00{\cdot}10^{-3}$	**87.94**	$7.78{\cdot}10^{-3}$	$8.19{\cdot}10^{-4}$	**89.47**	$1.24{\cdot}10^{3}$	$1.80{\cdot}10^{1}$	**98.55**
s13207a	$8.05{\cdot}10^{-3}$	$9.88{\cdot}10^{-4}$	**87.71**	$7.72{\cdot}10^{-3}$	$7.84{\cdot}10^{-4}$	**89.85**	$1.16{\cdot}10^{3}$	$1.74{\cdot}10^{1}$	**98.50**
s15850	$8.17{\cdot}10^{-3}$	$8.95{\cdot}10^{-4}$	**89.05**	$7.49{\cdot}10^{-3}$	$7.91{\cdot}10^{-4}$	**89.44**	$6.60{\cdot}10^{2}$	$1.77{\cdot}10^{1}$	**97.31**
s15850a	$7.22{\cdot}10^{-3}$	$8.64{\cdot}10^{-4}$	**88.03**	$6.69{\cdot}10^{-3}$	$7.33{\cdot}10^{-4}$	**89.05**	$6.37{\cdot}10^{2}$	$1.84{\cdot}10^{1}$	**97.11**
s35932	$2.14{\cdot}10^{-2}$	$3.00{\cdot}10^{-3}$	**85.96**	$1.96{\cdot}10^{-2}$	$2.60{\cdot}10^{-3}$	**86.75**	$6.32{\cdot}10^{2}$	$1.06{\cdot}10^{1}$	**98.31**
s38417	$2.23{\cdot}10^{-2}$	$2.45{\cdot}10^{-3}$	**89.00**	$2.16{\cdot}10^{-2}$	$2.30{\cdot}10^{-3}$	**89.37**	$1.87{\cdot}10^{3}$	$4.92{\cdot}10^{1}$	**97.36**
s38584	$1.89{\cdot}10^{-2}$	$2.37{\cdot}10^{-3}$	**87.48**	$1.81{\cdot}10^{-2}$	$2.06{\cdot}10^{-3}$	**88.64**	$1.83{\cdot}10^{3}$	$4.12{\cdot}10^{1}$	**97.75**
s38584a	$1.85{\cdot}10^{-2}$	$2.54{\cdot}10^{-3}$	**86.24**	$1.77{\cdot}10^{-2}$	$2.09{\cdot}10^{-3}$	**88.18**	$1.82{\cdot}10^{3}$	$3.99{\cdot}10^{1}$	**97.81**
b14	$6.18{\cdot}10^{-3}$	$9.74{\cdot}10^{-4}$	**84.22**	$5.57{\cdot}10^{-3}$	$7.18{\cdot}10^{-4}$	**87.11**	$2.20{\cdot}10^{3}$	$3.43{\cdot}10^{1}$	**98.44**
b14a	$4.93{\cdot}10^{-3}$	$6.94{\cdot}10^{-4}$	**85.91**	$4.46{\cdot}10^{-3}$	$4.92{\cdot}10^{-4}$	**88.96**	$1.37{\cdot}10^{3}$	$2.76{\cdot}10^{1}$	**97.98**
b15	$6.61{\cdot}10^{-3}$	$8.62{\cdot}10^{-4}$	**86.95**	$6.05{\cdot}10^{-3}$	$6.93{\cdot}10^{-4}$	**88.53**	$1.62{\cdot}10^{3}$	$3.40{\cdot}10^{1}$	**97.89**
b15a	$7.72{\cdot}10^{-3}$	$9.03{\cdot}10^{-4}$	**88.31**	$6.98{\cdot}10^{-3}$	$7.18{\cdot}10^{-4}$	**89.71**	$4.44{\cdot}10^{3}$	$1.10{\cdot}10^{2}$	**97.52**
b17	$2.25{\cdot}10^{-2}$	$2.59{\cdot}10^{-3}$	**88.50**	$2.16{\cdot}10^{-2}$	$2.09{\cdot}10^{-3}$	**90.32**	$1.23{\cdot}10^{4}$	$2.42{\cdot}10^{2}$	**98.02**
b17a	$2.45{\cdot}10^{-2}$	$2.68{\cdot}10^{-3}$	**89.07**	$2.34{\cdot}10^{-2}$	$2.17{\cdot}10^{-3}$	**90.71**	$1.54{\cdot}10^{4}$	$4.40{\cdot}10^{2}$	**97.15**
b18	$6.61{\cdot}10^{-2}$	$7.22{\cdot}10^{-3}$	**89.07**	$6.40{\cdot}10^{-2}$	$6.15{\cdot}10^{-3}$	**90.38**	$3.99{\cdot}10^{4}$	$1.34{\cdot}10^{3}$	**96.64**
b18a	$6.31{\cdot}10^{-2}$	$7.08{\cdot}10^{-3}$	**88.77**	$6.14{\cdot}10^{-2}$	$5.94{\cdot}10^{-3}$	**90.31**	$3.76{\cdot}10^{4}$	$1.24{\cdot}10^{3}$	**96.69**
b19	$1.37{\cdot}10^{-1}$	$1.46{\cdot}10^{-2}$	**89.31**	$1.32{\cdot}10^{-1}$	$1.26{\cdot}10^{-2}$	**90.45**	$9.58{\cdot}10^{4}$	$3.13{\cdot}10^{3}$	**96.73**
b19a	$1.31{\cdot}10^{-1}$	$1.37{\cdot}10^{-2}$	**89.55**	$1.27{\cdot}10^{-1}$	$1.23{\cdot}10^{-2}$	**90.29**	$9.20{\cdot}10^{4}$	$2.89{\cdot}10^{3}$	**96.85**
b20	$1.35{\cdot}10^{-2}$	$2.00{\cdot}10^{-3}$	**85.14**	$1.24{\cdot}10^{-2}$	$1.49{\cdot}10^{-3}$	**87.98**	$5.47{\cdot}10^{3}$	$1.04{\cdot}10^{2}$	**98.09**
b20a	$1.07{\cdot}10^{-2}$	$1.50{\cdot}10^{-3}$	**86.02**	$9.89{\cdot}10^{-3}$	$1.14{\cdot}10^{-3}$	**88.50**	$2.96{\cdot}10^{3}$	$5.39{\cdot}10^{1}$	**98.17**
b21	$1.35{\cdot}10^{-2}$	$2.03{\cdot}10^{-3}$	**84.99**	$1.25{\cdot}10^{-2}$	$1.51{\cdot}10^{-3}$	**87.89**	$5.56{\cdot}10^{3}$	$1.11{\cdot}10^{2}$	**98.00**
b21a	$1.07{\cdot}10^{-2}$	$1.50{\cdot}10^{-3}$	**85.99**	$9.87{\cdot}10^{-3}$	$1.12{\cdot}10^{-3}$	**88.67**	$3.06{\cdot}10^{3}$	$5.72{\cdot}10^{1}$	**98.12**
b22	$1.93{\cdot}10^{-2}$	$2.86{\cdot}10^{-3}$	**85.19**	$1.82{\cdot}10^{-2}$	$2.24{\cdot}10^{-3}$	**87.67**	$6.93{\cdot}10^{3}$	$1.53{\cdot}10^{2}$	**97.79**
b22a	$1.58{\cdot}10^{-2}$	$2.20{\cdot}10^{-3}$	**86.06**	$1.47{\cdot}10^{-2}$	$1.79{\cdot}10^{-3}$	**87.82**	$4.42{\cdot}10^{3}$	$9.21{\cdot}10^{1}$	**97.91**

Table A.19.: Peak and Average Test Power (TP) and Test Energy (TE) for Industrial Circuits (partially visualized in Fig. 10.7, Fig. 10.8 and Fig. 10.9).

	Test Power						**Test Energy**		
	Peak			**Average**					
Circuit *name* *(1)*	FTScan $[W]$ *(2)*	BFScan $[W]$ *(3)*	**Red. [%]** *(4)*	FTScan $[W]$ *(5)*	BFScan $[W]$ *(6)*	**Red. [%]** *(7)*	FTScan $[nJ]$ *(8)*	BFScan $[nJ]$ *(9)*	**Red. [%]** *(10)*
p35k	$3.05\cdot10^{-2}$	$3.52\cdot10^{-3}$	**88.46**	$2.75\cdot10^{-2}$	$3.09\cdot10^{-3}$	**88.74**	$2.03\cdot10^{4}$	$2.41\cdot10^{2}$	**98.80**
p45k	$3.40\cdot10^{-2}$	$5.01\cdot10^{-3}$	**85.26**	$3.19\cdot10^{-2}$	$3.82\cdot10^{-3}$	**88.02**	$3.43\cdot10^{4}$	$3.50\cdot10^{2}$	**98.97**
p77k	$5.08\cdot10^{-2}$	$5.73\cdot10^{-3}$	**88.72**	$4.61\cdot10^{-2}$	$5.00\cdot10^{-3}$	**89.15**	$1.28\cdot10^{4}$	$2.83\cdot10^{2}$	**97.78**
p78k	$5.89\cdot10^{-2}$	$9.62\cdot10^{-3}$	**83.66**	$5.77\cdot10^{-2}$	$8.56\cdot10^{-3}$	**85.16**	$2.54\cdot10^{3}$	$3.42\cdot10^{1}$	**98.65**
p81k	$6.13\cdot10^{-2}$	$8.02\cdot10^{-3}$	**86.92**	$5.91\cdot10^{-2}$	$7.13\cdot10^{-3}$	**87.94**	$1.16\cdot10^{4}$	$4.58\cdot10^{2}$	**96.05**
p89k	$6.24\cdot10^{-2}$	$6.94\cdot10^{-3}$	**88.87**	$5.98\cdot10^{-2}$	$6.12\cdot10^{-3}$	**89.76**	$2.46\cdot10^{4}$	$7.55\cdot10^{2}$	**96.93**
p100k	$8.08\cdot10^{-2}$	$9.21\cdot10^{-3}$	**88.59**	$7.81\cdot10^{-2}$	$8.33\cdot10^{-3}$	**89.33**	$8.22\cdot10^{4}$	$9.41\cdot10^{2}$	**98.85**
p141k	$1.47\cdot10^{-1}$	$2.13\cdot10^{-2}$	**85.54**	$1.45\cdot10^{-1}$	$1.93\cdot10^{-2}$	**86.66**	$5.07\cdot10^{4}$	$8.78\cdot10^{2}$	**98.26**
p239k	$2.53\cdot10^{-1}$	$2.93\cdot10^{-2}$	**88.41**	$2.49\cdot10^{-1}$	$2.68\cdot10^{-2}$	**89.24**	$7.54\cdot10^{4}$	$1.64\cdot10^{3}$	**97.82**
p259k	$2.62\cdot10^{-1}$	$3.73\cdot10^{-2}$	**85.77**	$2.47\cdot10^{-1}$	$3.27\cdot10^{-2}$	**86.76**	$9.11\cdot10^{4}$	$2.09\cdot10^{3}$	**97.69**
p267k	$2.27\cdot10^{-1}$	$2.41\cdot10^{-2}$	**89.37**	$2.23\cdot10^{-1}$	$2.17\cdot10^{-2}$	**90.28**	$1.04\cdot10^{5}$	$1.67\cdot10^{3}$	**98.39**
p269k	$2.27\cdot10^{-1}$	$2.41\cdot10^{-2}$	**89.37**	$2.22\cdot10^{-1}$	$2.19\cdot10^{-2}$	**90.12**	$1.05\cdot10^{5}$	$1.68\cdot10^{3}$	**98.40**
p279k	$2.31\cdot10^{-1}$	$2.51\cdot10^{-2}$	**89.12**	$2.26\cdot10^{-1}$	$2.38\cdot10^{-2}$	**89.48**	$1.01\cdot10^{5}$	$1.94\cdot10^{3}$	**98.08**
p286k	$2.73\cdot10^{-1}$	$3.56\cdot10^{-2}$	**86.94**	$2.44\cdot10^{-1}$	$3.12\cdot10^{-2}$	**87.20**	$1.46\cdot10^{5}$	$4.18\cdot10^{3}$	**97.14**
p295k	$2.49\cdot10^{-1}$	$2.52\cdot10^{-2}$	**89.89**	$2.34\cdot10^{-1}$	$2.30\cdot10^{-2}$	**90.14**	$2.22\cdot10^{5}$	$4.77\cdot10^{3}$	**97.84**
p330k	$2.64\cdot10^{-1}$	$2.67\cdot10^{-2}$	**89.88**	$2.59\cdot10^{-1}$	$2.49\cdot10^{-2}$	**90.38**	$3.04\cdot10^{5}$	$3.29\cdot10^{3}$	**98.91**
p378k	$2.96\cdot10^{-1}$	$4.58\cdot10^{-2}$	**84.52**	$2.91\cdot10^{-1}$	$4.11\cdot10^{-2}$	**85.85**	$1.25\cdot10^{4}$	$2.31\cdot10^{2}$	**98.15**
p388k	$3.75\cdot10^{-1}$	$4.79\cdot10^{-2}$	**87.22**	$3.37\cdot10^{-1}$	$4.44\cdot10^{-2}$	**86.80**	$1.05\cdot10^{5}$	$2.70\cdot10^{3}$	**97.43**
p418k	$3.87\cdot10^{-1}$	$3.97\cdot10^{-2}$	**89.74**	$3.82\cdot10^{-1}$	$3.77\cdot10^{-2}$	**90.12**	$1.88\cdot10^{5}$	$4.67\cdot10^{3}$	**97.51**
p469k	$2.43\cdot10^{-2}$	$5.39\cdot10^{-3}$	**77.79**	$2.34\cdot10^{-2}$	$4.03\cdot10^{-3}$	**82.78**	$4.84\cdot10^{3}$	$7.75\cdot10^{1}$	**98.39**
p483k	$4.59\cdot10^{-1}$	$6.10\cdot10^{-2}$	**86.71**	$4.47\cdot10^{-1}$	$5.61\cdot10^{-2}$	**87.44**	$8.83\cdot10^{4}$	$4.48\cdot10^{3}$	**94.92**
p500k	$4.13\cdot10^{-1}$	$4.25\cdot10^{-2}$	**89.70**	$4.06\cdot10^{-1}$	$3.65\cdot10^{-2}$	**91.02**	$4.95\cdot10^{5}$	$8.83\cdot10^{3}$	**98.21**
p533k	$4.83\cdot10^{-1}$	$6.83\cdot10^{-2}$	**85.86**	$4.27\cdot10^{-1}$	$5.72\cdot10^{-2}$	**86.61**	$1.26\cdot10^{5}$	$4.86\cdot10^{3}$	**96.14**
p874k	$5.56\cdot10^{-1}$	$5.64\cdot10^{-2}$	**89.85**	$5.48\cdot10^{-1}$	$5.03\cdot10^{-2}$	**90.81**	$3.64\cdot10^{5}$	$6.03\cdot10^{3}$	**98.34**

Index

ATPG, *see* test pattern generation

bathtub curve, 3
benchmark circuits, 180
    industrial
        NXP, 181
    public
        ISCAS89, 181
        ITC99, 181
BISER, 46

characteristic
    extended, 106
    modulo-2 address, 43, 68
    recomputed C', 68
    reference, 43
    reference C, 68
    tree, 71
        extended, 106
charge
    collected, 9
    collection, 8
    critical, 9
circuit, 23, 24
    abstraction level, 23
        algorithmic, 24
        architectural/system, 24
        circuit/transistor, 24
        functional block/register transfer, 24
        logical/gate, 24
    combinational, 24
        definition, 25
    domain of description
        behavioral, 24
        physical, 24
        structural, 24
    output function, 26
    sequential, 26
        combinational core, 26
        edge-triggered, 26
        level-sensitive, 26
        state, 26
    transition function, 26
CMOS, *see* complementary metal oxide semiconductor
CNF, *see* conjunctive normal form
complementary metal oxide semiconductor, 3

DED, *see* Single and Double Error Detection
defect, 2, 26
    definition, 26
derating
    logic, 29
    time, 29

design for test, 12

EDAC, *see* error detection and correction
EDT, *see* embedded deterministic test
embedded deterministic test, 51
error, 2, 27
    correction, 30
    definition, 27
    detection, 30
    localization, 30
    soft error, 2, 10
error correcting code, 16, 41, 42
error detection and correction, 16, 31

failure, 2, 27
    definition, 27
    rate, 3
failure mechanism, 4
    extrinsic, 4
        random, 4
        systematic, 5
    intrinsic, 5
fault, 12, 26
    classification
        detected, 38
        undetectable, 38
        undetected, 38
    coverage, 37
    definition, 26
    efficiency, 38
    hard fault, 2, 10
    location, 12
    model, *see* fault model
    non-permanent, 27
        intermittent, 27
        transient, 28
    permanent, 27
    simulation, 37
fault model, 12, 33
    delay, 33
        gate, 33
        path, 33
    gross delay/transition, 33, 34
    small delay, 33
    structural, 33
    stuck-at, 12, 34
fault tolerance, 30, 60
    correction, 30, 85
    detection, 30, 61
    localization, 30, 61

GRAAL, 49

Hamming
    code, 32
    distance
        minimum, 31
    relationship, 32
hardware description language, 24

infrastructure, 60

masking
    electrical, 11
    logical, 11
    temporal, 11
MBU, *see* Multiple Bit Upset
MCU, *see* Multiple Cell Upset
MTTF, *see* mean time to failure

performance, 1

RAZOR, 48

RAZOR-II, 50
RAZOR-Lite, 50
redundancy, 30
  information, 30
  structural, 30
  temporal, 30
reliability, 2

SAT, *see* satisfiability
satisfiability, 39
  assumption, 40, 138
  conjunctive normal form, 39, 133
  incremental solving, 40, 138
  instance, 39
  model, 39
  satisfiable, 39
  unsatisfiable, 39
SBU, *see* Single Bit Upset
scan design, 13, 35
  edge-triggered, 35
  level sensitive, 35
  scan chain, 13, 35
SEC, *see* Single Error Correction
SECDED, *see* Single Error Correction and Double Error Detection
SED, *see* Single Error Detection
SEE, *see* Single Event Effect
SER, *see* soft error rate
SEU, *see* Single Event Upset
Single Error Correction, 32, 90
  and Double Error Detection, 32, 42, 108
Single Error Detection, 31, 87
  and Double Error Detection, 108
Single Event Effect, 2, 10
  Single Event Transient, 11
  Single Event Upset, 11, 28, 61, 85, 103
    Multiple Bit Upset, 11, 16, 29, 103
    Multiple Cell Upset, 11, 29, 103
    Single Bit Upset, 11, 16, 29
soft error, 28
  classification, 10
  mitigation, 15
    design, 15
    process technology, 15
    source level, 15
    system level redundancy, 16
  quantification, 29
    failures in time, 29
    mean time to failure, 30
    soft error rate, 29
  source, 7
    alpha particle, 8
    high-energy neutron, 7
    low-energy neutron, 7
soft error mitigation, 41
  memory, 41
    error correcting code, 42
    error detecting refreshment, 43
  sequential elements, 45
    BISER, 46
    GRAAL, 49
    RAZOR, 48
    RAZOR-II, 50
    RAZOR-Lite, 50
standard cell
  custom

Bit-Flipping Latch (BFL), 91, 96
Parity-Pair Latch (PPL), 66, 76
Transmission-Gate Exclusive OR (TGXOR), 114, 116
library
Open Cell Library (OCL), 74, 93, 108, 115, 141

test, 2, 12, 33
automatic test equipment (ATE), 12
built-in self (BIST), 12
cost, 14
escape, 12
external, 12
functional, 12
in-field, 12
manufacturing, 12
metric
area overhead, 14
average test power, 14, 37
peak test power, 14, 37
test application time, 14, 36
test data volume, 13, 14, 37
test energy, 14, 37
testability, 13
pass/fail, 12
pattern, 12
generation, 38
response, 12
sequence
generation, 131
set, 12
structural, 12
tester, 12
test access, 51
random access scan, 53
progressive, 55
toggle, 55
test data compaction, 51
test data compression, 51
test data compression and compaction, 51
embedded deterministic test, 51
X-compact, 53
test data
compaction, 51
convolutional, 53
space, 53
time, 52
compression, 51
broadcast scan, 52
code, 52
linear, 52
testability, 13
controllability, 13
observability, 13
TMR, *see* Triple Modular Redundancy
transmission-gate, 66, 91, 114
Triple Modular Redundancy, 46
Tseitin transformation, 133

vulnerability factor
architectural, 29
time, 29

yield, 1

Curriculum Vitae of the Author

Michael E. Imhof received a Diploma degree in Computer Science (Diplom-Informatik) in 2005 from the University of Stuttgart, where he joined the Institute of Computer Architecture and Computer Engineering (Institut für Technische Informatik, ITI) in 2006.

The author has been working as a research and teaching assistant under the guidance of Prof. Dr. rer. nat. habil. H.-J. Wunderlich and supported the courses “Computer Architecture”, “Hardware Lab”, “Algorithms and Methods for Electronic Design Automation of Nano- and Microelectronics”, and “Design and Implementation of Electronic Design Automation Tools”. He promoted over twenty young academics in several seminars, undergraduate projects as well as Diploma and Master theses.

He has been involved in the research projects “Improved Testing of VLSI Chips with Power Constraints” funded by IBM Research & Development and “Combining Fault Tolerance and Offline Test Strategies for Nanoscaled Electronics” funded by the German Academic Exchange Service (DAAD). Furthermore, he contributed to four projects funded by the German Research Foundation (DFG): “REALTEST - Test and Reliability of Nanoelectronic Systems”, “OTERA - Online Test Strategies for Reliable Reconfigurable Architectures” within the DFG Priority Program SPP 1500, “ROCK - Robust Network-On-Chip-Communication Through Hierarchical Online Diagnosis and Reconfiguration”, and “RM-BIST - Reliability Monitoring and Managing Built-In Self Test”.

He is co-recipient of the 2008 Best Paper Award at the IEEE International Symposium on Electronic Design, Test and Applications, the 2014 Best Paper Award at the IEEE European Test Symposium, and a 2014 ’European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC)’ Paper Award. His research interests include low-power test, variation-aware test as well as fault tolerance.

Publications of the Author

Journal Articles (refereed)

[J3] S. Holst, M. E. Imhof, and H.-J. Wunderlich, “High-Throughput Logic Timing Simulation on GPGPUs”, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 3, pp. 1–22, Jun. 2015. DOI: 10.1145/2714564.

[J2] L. Bauer, C. Braun, M. E. Imhof, M. A. Kochte, E. Schneider, H. Zhang, J. Henkel, and H.-J. Wunderlich, “Test Strategies for Reliable Runtime Reconfigurable Architectures”, *IEEE Transactions on Computers*, vol. 62, no. 8, pp. 1494–1507, Aug. 2013. DOI: 10.1109/TC.2013.53.

[J1] R. Baranowski, S. Di Carlo, N. Hatami, M. E. Imhof, M. A. Kochte, P. Prinetto, H.-J. Wunderlich, and C. G. Zoellin, “Efficient Multi-level Fault Simulation of HW/SW Systems for Structural Faults”, *SCIENCE CHINA Information Sciences*, vol. 54, no. 9, pp. 1784–1796, Sep. 2011. DOI: 10.1007/s11432-011-4366-9.

Conference Proceedings (refereed formal proceedings)

[C27] A. Dalirsani, N. Hatami, M. E. Imhof, M. Eggenberger, G. Schley, M. Radetzki, and H.-J. Wunderlich, “On Covering Structural Defects in NoCs by Functional Tests”, in *Proc. 23rd IEEE Asian Test Symposium (ATS)*, Hangzhou, China, Nov. 2014, pp. 87–92. DOI: 10.1109/ATS.2014.27.

[C26] H. Zhang, M. A. Kochte, M. E. Imhof, L. Bauer, H.-J. Wunderlich, and J. Henkel, “GUARD: GUAranteed Reliability in Dynamically Reconfigurable Systems”, in *Proc. 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, HiPEAC Paper Award, San Francisco, CA, USA, Jun. 2014, pp. 1–6. DOI: 10.1145/2593069.2593146.

[C25] M. Sauer, I. Polian, M. E. Imhof, A. Mumtaz, E. Schneider, A. Czutro, H.-J. Wunderlich, and B. Becker, "Variation-Aware Deterministic ATPG", in *Proc. 19th IEEE European Test Symposium (ETS)*, Best Paper Award, Paderborn, Germany, May 2014, pp. 87–92. DOI: 10.1109/ETS.2014.6847806.

[C24] A. Dalirsani, M. E. Imhof, and H.-J. Wunderlich, "Structural Software-Based Self-Test of Network-on-Chip", in *Proc. 32nd IEEE VLSI Test Symposium (VTS)*, Napa, CA, USA, Apr. 2014, pp. 1–6. DOI: 10.1109/VTS.2014.6818754.

[C23] M. E. Imhof and H.-J. Wunderlich, "Bit-Flipping Scan - A Unified Architecture for Fault Tolerance and Offline Test", in *Proc. Design, Automation and Test in Europe (DATE)*, Dresden, Germany, Mar. 2014, pp. 1–6. DOI: 10.7873/DATE.2014.206.

[C22] R. Baranowski, A. Cook, M. E. Imhof, C. Liu, and H.-J. Wunderlich, "Synthesis of Workload Monitors for On-Line Stress Prediction", in *Proc. 16th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, New York City, NY, USA, Oct. 2013, pp. 137–142. DOI: 10.1109/DFT.2013.6653596.

[C21] H. Zhang, L. Bauer, M. A. Kochte, E. Schneider, C. Braun, M. E. Imhof, H.-J. Wunderlich, and J. Henkel, "Module Diversification: Fault Tolerance and Aging Mitigation for Runtime Reconfigurable Architectures", in *Proc. IEEE International Test Conference (ITC)*, Anaheim, CA, USA, Sep. 2013, pp. 1–10. DOI: 10.1109/TEST.2013.6651926.

[C20] A. Czutro, M. E. Imhof, J. Jiang, A. Mumtaz, M. Sauer, B. Becker, I. Polian, and H.-J. Wunderlich, "Variation-Aware Fault Grading", in *Proc. 21st IEEE Asian Test Symposium (ATS)*, Niigata, Japan, Nov. 2012, pp. 344–349. DOI: 10.1109/ATS.2012.14.

[C19] M. S. Abdelfattah, L. Bauer, C. Braun, M. E. Imhof, M. A. Kochte, H. Zhang, J. Henkel, and H.-J. Wunderlich, "Transparent Structural Online Test for Reconfigurable Systems", in *Proc. 18th IEEE International On-Line Testing Symposium (IOLTS)*, Sitges, Spain, Jun. 2012, pp. 37–42. DOI: 10.1109/IOLTS.2012.6313838.

[C18] L. Bauer, C. Braun, M. E. Imhof, M. A. Kochte, H. Zhang, H.-J. Wunderlich, and J. Henkel, "OTERA: Online Test Strategies for Reliable Reconfigurable Architectures", in *Proc. NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Nuremberg, Germany, Jun. 2012, pp. 38–45. DOI: 10.1109/AHS.2012.6268667.

[C17] D. A. Tran, A. Virazel, A. Bosio, L. Dilillo, P. Girard, A. Todri, M. E. Imhof, and H.-J. Wunderlich, "A Pseudo-Dynamic Comparator for Error Detection in Fault Tolerant Architectures", in *Proc. 30th IEEE VLSI Test Symposium (VTS)*, Maui, HI, USA, Apr. 2012, pp. 50–55. DOI: 10.1109/VTS.2012.6231079.

[C16] A. Cook, S. Hellebrand, M. E. Imhof, A. Mumtaz, and H.-J. Wunderlich, "Built-in Self-Diagnosis Targeting Arbitrary Defects with Partial Pseudo-Exhaustive Test", in *Proc. 13th IEEE Latin-American Test Workshop (LATW)*, Quito, Ecuador, Apr. 2012, pp. 1–4. DOI: 10.1109/LATW.2012.6261229.

[C15] A. Mumtaz, M. E. Imhof, S. Holst, and H.-J. Wunderlich, "Embedded Test for Highly Accurate Defect Localization", in *Proc. 20th IEEE Asian Test Symposium (ATS)*, New Delhi, India, Nov. 2011, pp. 213–218. DOI: 10.1109/ATS.2011.60.

[C14] A. Mumtaz, M. E. Imhof, S. Holst, and H.-J. Wunderlich, "Eingebetteter Test zur hochgenauen Defekt-Lokalisierung", in *Proc. 5. GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf (ZuE)*, Hamburg-Harburg, Germany, Sep. 2011, pp. 43–47, ISBN: 978-3-8007-3357-6.

[C13] M. E. Imhof and H.-J. Wunderlich, "Korrektur transienter Fehler in eingebetteten Speicherelementen", in *Proc. 5. GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf (ZuE)*, Hamburg-Harburg, Germany, Sep. 2011, pp. 76–83, ISBN: 978-3-8007-3357-6.

[C12] A. Mumtaz, M. E. Imhof, and H.-J. Wunderlich, "P-PET: Partial Pseudo-Exhaustive Test for High Defect Coverage", in *Proc. IEEE International Test Conference (ITC)*, Anaheim, CA, USA, Sep. 2011, pp. 1–8. DOI: 10.1109/TEST.2011.6139130.

[C11] M. E. Imhof and H.-J. Wunderlich, "Soft Error Correction in Embedded Storage Elements", in *Proc. 17th IEEE International On-Line Testing Symposium (IOLTS)*, Athens, Greece, Jul. 2011, pp. 169–174. DOI: 10.1109/IOLTS.2011.5993832.

[C10] M. A. Kochte, C. G. Zoellin, R. Baranowski, M. E. Imhof, H.-J. Wunderlich, N. Hatami, S. Di Carlo, and P. Prinetto, "Efficient Simulation of Structural Faults for the Reliability Evaluation at System-Level", in *Proc. 19th IEEE Asian Test Symposium (ATS)*, Shanghai, China, Dec. 2010, pp. 3–8. DOI: 10.1109/ATS.2010.10.

[C9] M. A. Kochte, C. G. Zoellin, R. Baranowski, M. E. Imhof, H.-J. Wunderlich, N. Hatami, S. Di Carlo, and P. Prinetto, "System Reliability Evaluation Using Concurrent Multi-Level Simulation of Structural Faults", in *Proc. IEEE International Test Conference (ITC)*, Austin, TX, USA, Oct. 2010, p. 1. DOI: 10.1109/TEST.2010.5699309.

[C8] M. A. Kochte, C. G. Zoellin, R. Baranowski, M. E. Imhof, H.-J. Wunderlich, N. Hatami, S. Di Carlo, and P. Prinetto, "Effiziente Simulation von strukturellen Fehlern für die Zuverlässigkeitsanalyse auf Systemebene", in *Proc. 4. GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf (ZuE)*, Wildbad Kreuth, Germany, Sep. 2010, pp. 25–32, ISBN: 978-3-8007-3299-9.

[C7] M. A. Kochte, C. G. Zoellin, M. E. Imhof, R. Salimi Khaligh, M. Radetzki, H.-J. Wunderlich, S. Di Carlo, and P. Prinetto, "Test Exploration and Validation Using Transaction Level Models", in *Proc. Design, Automation and Test in Europe (DATE)*, Nice, France, Apr. 2009, pp. 1250–1253. DOI: 10.1109/DATE.2009.5090856.

[C6] M. E. Imhof, H.-J. Wunderlich, and C. G. Zoellin, "Erkennung von transienten Fehlern in Schaltungen mit reduzierter Verlustleistung", in *Proc. 2. GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf (ZuE)*, Ingolstadt, Germany, Sep. 2008, pp. 107–114, ISBN: 978-3-8007-3119-0.

[C5] M. E. Imhof, H.-J. Wunderlich, and C. G. Zoellin, "Integrating Scan Design and Soft Error Correction in Low-Power Applications", in *Proc. 14th IEEE International On-Line Testing Symposium (IOLTS)*, Rhodes, Greece, Jul. 2008, pp. 59–64. DOI: 10.1109/IOLTS.2008.31.

[C4] M. Elm, H.-J. Wunderlich, M. E. Imhof, C. G. Zoellin, J. Leenstra, and N. Maeding, "Scan chain clustering for test power reduction", in *Proc. 45th ACM/IEEE Design Automation Conference (DAC)*, Anaheim, CA, USA, Jun. 2008, pp. 828–833. DOI: 10.1145/1391469.1391680.

[C3] M. A. Kochte, C. G. Zoellin, M. E. Imhof, and H.-J. Wunderlich, "Test Set Stripping Limiting the Maximum Number of Specified Bits", in *Proc. 4th IEEE International Symposium on Electronic Design, Test and Applications (DELTA)*, Best Paper Award, Hong Kong, China, Jan. 2008, pp. 581–586. DOI: 10.1109/DELTA.2008.64.

[C2] M. E. Imhof, C. G. Zoellin, H.-J. Wunderlich, N. Maeding, and J. Leenstra, "Scan Test Planning for Power Reduction", in *Proc. 44th ACM/IEEE Design Automation Conference (DAC)*, San Diego, CA, USA, Jun. 2007, pp. 521–526. DOI: 10.1145/1278480.1278614.

[C1] M. E. Imhof, C. G. Zoellin, H.-J. Wunderlich, N. Maeding, and J. Leenstra, "Verlustleistungsoptimierende Testplanung zur Steigerung von Zuverlässigkeit und Ausbeute", in *Proc. 1. GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf (ZuD)*, Munich, Germany, Mar. 2007, pp. 69–76, ISBN: 978-3-8007-3023-0.

Workshop Contributions (refereed)

[W4] A. Mumtaz, M. E. Imhof, and H.-J. Wunderlich, "Mixed-Mode-Mustererzeugung für hohe Defekterfassung beim Eingebetteten Test", in *Proc. 23rd GI/GMM/ITG Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ)*, Passau, Germany, Feb. 2011, pp. 55–58.

[W3] M. A. Kochte, C. G. Zoellin, M. E. Imhof, R. Salimi Khaligh, M. Radetzki, H.-J. Wunderlich, S. Di Carlo, and P. Prinetto, "Modellierung der Testinfrastruktur auf der Transaktionsebene", in *Proc. 21st ITG/GI/GMM Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ)*, Bremen, Germany, Feb. 2009, pp. 61–66.

[W2] M. E. Imhof, H.-J. Wunderlich, and C. G. Zoellin, "Integrating Scan Design and Soft Error Correction in Low-Power Applications", in *Proc. 1st Workshop on Low Power Design Impact on Test and Reliability (LPonTR)*, Verbania, Italy, May 2008, pp. 14–16.

[W1] M. E. Imhof, H.-J. Wunderlich, C. G. Zoellin, J. Leenstra, and N. Maeding, "Reduktion der Verlustleistung beim Selbsttest durch Verwendung testmengenspezifischer Information", in *Proc. 20th ITG/GI/GMM Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ)*, Wien, Austria, Feb. 2008, pp. 137–141.

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

Michael E. Imhof