Dependable Reconfigurable Scan Networks

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

Natalia Lylina

aus Moskau, Russland

Hauptberichter: Prof. Hans-Joachim Wunderlich
Mitberichter: Prof. Matteo Sonza Reorda

Tag der mündlichen Prüfung: 16. Dezember 2022.

Institut für Technische Informatik der Universität Stuttgart

2022

Contents

Acknowledgments ix

Abstract xi

Zusammenfassung xiii

List of Figures xv

List of Tables xix

List of Abbreviations xxi

List of Symbols xxiv

1 Introduction 1

- 1.1 System Dependability Management . . . 1
  - 1.1.1 Dependability Requirements of the Modern Devices . . . 1
  - 1.1.2 Dependability-Enhancing Means . . . 4
  - 1.1.3 Dependability Management throughout the Lifetime . . . 5
- 1.2 Reconfigurable Scan Networks (RSNs) for Dependability Enhancement . . 8
  - 1.2.1 Dependability Management Support via RSNs . . . 8
  - 1.2.2 Dependability Issues due to Improper RSN Integration . . . 9
- 1.3 Overview of the Work . . . 11
  - 1.3.1 Major Contributions . . . 11
  - 1.3.2 Document Organization . . . 13
- 1.4 Summary . . . 16

I Background and State of the Art

2 Reconfigurable Scan Networks 20

2.1 RSN Structure 20
2.2 RSN Operation 22
2.3 Faults in RSNs 24
  2.3.1 Faults at Interfaces to Instruments 24
  2.3.2 Faults in Control Primitives 25
  2.3.3 Faults in Scan Segments 26
2.4 Graph-Based Modeling of RSNs 26
  2.4.1 Basic Definitions 27
  2.4.2 RSN Graph 28
2.5 Summary 30

3 State of the Art 33

3.1 RSNs and their Applications 33
3.2 Robust Access via RSNs 34
  3.2.1 Fault Removal 34
  3.2.2 Fault Tolerance 34
3.3 Test and Diagnosis of RSNs 35
  3.3.1 Test of Scan Interfaces 36
  3.3.2 Test of Control Primitives 36
  3.3.3 Test of Scan Segments 37
3.4 Security Compliance of RSNs 38
  3.4.1 Security Analysis and Specification 38
  3.4.2 Security-Oriented Integration of Scan Chains 39
3.5 Summary 40

II Dependability-Oriented Integration of Reconfigurable Scan Networks

4 Robust RSNs 45

4.1 Goals of Robustness-Enhancing Resynthesis 46
  4.1.1 Optimization Criteria 47
4.2 Criticality Analysis 51
  4.2.1 Series-Parallel RSNs 52
  4.2.2 Fault Effects in Terms of Graph Connectivity 54
  4.2.3 Criticality Analysis for Series-Parallel RSNs 56
4.3 Selective Hardening for Robustness Enhancement 57
  4.3.1 Hardening as a Multi-Objective Problem 57
  4.3.2 Evolutionary Approach for Primitives Selection 58
4.4 Application for Arbitrary RSN Structures 60
  4.4.1 Validation of the Series-Parallel Property 61
  4.4.2 Transformation into a Series-Parallel Graph 61
4.5 Evaluation 64
  4.5.1 Series-Parallel Property 64
  4.5.2 Robustness Enhancement 65
4.6 Summary 68

5 Testable RSNs **70**
5.1 A Complete Design-for-Test Scheme 71
  5.1.1 Optimization Criteria 72
  5.1.2 DfT as Essential Companion for RSN Test 73
  5.1.3 Test of the RSN Interfaces 75
  5.1.4 Test of Control Primitives 78
  5.1.5 Test of Scan Segments 85
  5.1.6 Test Integration 88
5.2 Extension for Online Periodic Test of RSNs 90
  5.2.1 Test Generation Method Overview 91
  5.2.2 Scheduling of Test Accesses 93
  5.2.3 Selection of Relevant Active Scan Paths 97
  5.2.4 SP-RSN Path Selection 97
  5.2.5 Non-SP-RSN Path Selection 98

5.3 Evaluation . . . 99
  5.3.1 Design-for-Test Scheme . . . 99
  5.3.2 Online Periodic Test . . . 103
5.4 Summary . . . 106

6 Security Compliant RSNs 108
6.1 Security Compliance Analysis of RSNs . . . 109
  6.1.1 Security Specifications . . . 109
  6.1.2 Reachability Analysis of RSNs . . . 111
  6.1.3 Hybrid Connectivities Computation . . . 115
  6.1.4 Identification of Violating Connectivities . . . 116
6.2 Synthesis of Security Compliant RSNs . . . 117
  6.2.1 Structural Security-Compliance Aware Resynthesis . . . 117
  6.2.2 Hybrid Protection Method for Multiple Users . . . 128
  6.2.3 General Protection Flow . . . 135
6.3 Evaluation . . . 138
  6.3.1 Structural Resynthesis for Single User Protection . . . 138
  6.3.2 Hybrid Protection Scheme for Multiple Users . . . 142
6.4 Summary . . . 144

7 Integration of Dependable RSNs 146
7.1 Complete Solution for Resynthesis of Dependable RSNs . . . 146
  7.1.1 Structural Changes to the RSN Topology . . . 148
  7.1.2 Changes to the Segments Lengths . . . 149
  7.1.3 Changes to the Properties of Scan Primitives . . . 149
  7.1.4 Dependable Access Interface . . . 150
  7.1.5 Compatibility of the Applied Changes . . . 151
7.2 Summary . . . 152

III Conclusions and Future Work

8 Conclusions 155

8.1 General Conclusions 155
8.2 Ongoing and Future Work 157

Appendices **159**

A Experimental Setup **161**

A.1 Benchmarks 161
  A.1.1 RSN Benchmarks 161
  A.1.2 Connectivities between the Instruments 161
A.2 Electronic Design Automation (EDA) and other Tools 162

B Dependability Enhancement Results **164**

B.1 Robust RSNs 164
B.2 Testable RSNs 166
  B.2.1 Control Lines 166
  B.2.2 Test Integration 168
  B.2.3 Online Periodic Test 170
B.3 Security Compliant RSNs 173
  B.3.1 Structural Resynthesis for Single User Protection 173
  B.3.2 Hybrid Protection Scheme for Multiple Users 176

Bibliography **181**

Index **193**

Curriculum Vitae of the Author **195**

Publications of the Author **199**

Acknowledgments

Words cannot express my gratitude to my supervisor, Prof. Hans-Joachim Wunderlich, for all his help throughout this long and challenging research trip. This journey would never be successful without his patient belief in me. Throughout my studies, I had a unique opportunity to have a lot of fruitful and challenging research discussions with him, which taught me how to come up with and develop new ideas. I would also like to sincerely thank Prof. Matteo Sonza Reorda for taking over the role as my second adviser and for having motivating research discussions during the conferences and seminars. It is a great honor for me to have these two pioneers of the semiconductor test field as advisors for this doctoral thesis.

My sincere thanks also address my dear colleagues at the Institut für Technische Informatik (ITI) of Stuttgart University, both at the RA department (Ahmed, Alexander, Alexia, Chang, Claus, Eric, Hanieh, Paria and Peter), as well as to my dear colleagues from the HOCOS group of Prof. Ilia Polian (especially, Devanshi, Florian, Nourhan, Denis, Sebastian, Roshwin and Mael) and the STAR group of Prof. Hussam Amrouch (especially, Florian and Paul). I am grateful to my colleagues Ahmed and Paria for starting this journey with me and going together throughout the sweet and bitter parts of the Ph.D. student life. I would also like to thank Chang Liu and Eric Schneider for being not only dear colleagues, which helped me find a path in the research community through their comments and critics, but also excellent friends. My journey would not be possible without my dear co-author and close friend Peter Wang, who was always there to discuss research and personal problems. I want to thank my dear friend Mirjam for organizing the work at ITI in a smooth and friendly manner, and for many hours at the coffee corner. My special thanks go to Lothar and Helmut - under their careful guidance, the university felt like a second home for me.

The collaborations and research discussions within the University of Stuttgart and above have helped me a lot. I am grateful to all collaborators within the Graduate School for Test and Reliability at the University of Stuttgart.

I also express my gratitude to the groups of Prof. Bernd Becker and Prof. Sybille Hellebrand. It applies, among others, to Matthias Sauer, who helped me a lot not only to understand some research concepts but to gain more self-confidence. It also applies to Somayeh Kohan, Jan Dennis Reimer and Alexander Sprenger, who enhanced my research works through fruitful discussions and made the conference attendance much more entertaining.

My journey would not be possible without my friends and family. My most sincere thanks go to my parents, who raised me with lots of love and care. Their careful support throughout my life allowed me to concentrate on the things, that matter to me, and to freely develop my skills. Together with my husband, Slava, they are my most solid background and my cozy home, wherever I am. I would also like to thank Slava for patiently staying with me throughout all the deadlines, the notifications of acceptance, and the impostor syndrome moments. His belief in me has helped me not to give up and kept my spirits and motivation high during this process.

My friends, Olga and Sergey, thank you for moving to Stuttgart with us and being there throughout all the happy and sad moments of our lives. My school friends, Jenia and Lena, and my university friends, Tonia and Natascha, thank you for being there for me despite the long distances in between. I would also like to thank my advisors during my master's studies Arkadi Poliakov, Galina Kalinina, Alexander Kriukov, and the entire research group in the Moscow Power Engineering Institute for introducing me to hardware design.

Lastly, I would be remiss in not mentioning other family members, including my dear grandmother and my beloved auntie. I am so sorry, that my other grandparents passed away too early and they could not see me now, but I would always be thankful for them for their belief in me. Overall, I would like to thank all the people I collaborated with and am befriended, for showing me that both science and friendship are not aware of such concepts as "nationality".

Stuttgart, September 2022
Natalia Lylina

Abstract

The dependability of modern devices is enhanced by integrating an extensive number of extra-functional instruments. These are needed to facilitate cost-efficient bring-up, debug, test, diagnosis, and adaptivity in the field and might include, e.g., sensors, aging monitors, Logic, and Memory Built-In Self-Test (BIST) registers. Reconfigurable Scan Networks (RSNs) provide a flexible way to access such instruments as well the device's registers throughout the lifetime, starting from post-silicon validation (PSV) through manufacturing test and finally during in-field operation. At the same time, the dependability properties of the system can be affected through an improper RSN integration.

This doctoral project overcomes these problems and establishes a methodology to integrate dependable RSNs for a given system considering the most relevant dependability aspects, such as robustness, testability, and security compliance of RSNs.

Zusammenfassung

Die Zuverlässigkeit moderner integrierter Systeme wird durch die Integration einer Vielzahl von extra-funktionalen Instrumenten erhöht. Diese werden benötigt, um ein System kosteneffizient zu starten, debuggen, testen, diagnostizieren und im Feld einzupassen. Sie können beispielsweise Sensoren, Alterungsmonitore, Logik- und Speicher-integrierte Selbsttest (BIST) Register umfassen. Rekonfigurierbare Scan-Netzwerke (RSNs) bieten eine flexible Möglichkeit, auf solche Instrumente sowie auf interne Register des Systems während der gesamten Lebensdauer zuzugreifen, angefangen bei der Post-Silicon Validierung (PSV) über den Herstellungstest bis hin zur Feldoperation. Gleichzeitig können aber die Zuverlässigkeitseigenschaften des Systems durch eine unsachgemäße RSN-Integration beeinträchtigt werden.

Die vorliegende Doktorarbeit behandelt diese Probleme und etabliert eine Methodik zur Integration zuverlässiger RSNs für ein gegebenes System unter Berücksichtigung solcher Zuverlässigkeitsaspekte, wie Zugänglichkeit über RSNs, Testbarkeit von RSNs und Sicherheitskonformität von RSNs mit dem zugrunde liegenden System.

List of Figures

Chapter 1: Introduction 1

1.1 RSN accessing the instruments 8
1.2 Objectives and Contributions 12

Chapter 2: Reconfigurable Scan Networks 20

2.1 Running RSN example 21
2.2 Scan segment internals 21
2.3 SIB transformation 22
2.4 Scan cell accesses an instrument 24
2.5 A scan cell with injected faults 25
2.6 Graph traversal notations 27
2.7 Example graph of an RSN from Fig. 2.1 28

Chapter 4: Robust RSNs 45

4.1 Hardened RSN Example 46
4.2 Pareto front 48
4.3 Series-parallel property examples 53
4.4 Binary decomposition tree for the graph in Fig. 2.7 54
4.5 Faulty tree from Fig. 4.4, the SIB is affected by a stuck-at-1 fault 56
4.6 Tree from Fig. 4.4 with merged nodes 57
4.7 Non series-parallel graph 62
4.8 Binary decomposition tree for the non-series-parallel RSN graph in Fig. 4.7 63

4.9 Transformed subgraph of the non-series-parallel graph from Fig. 4.7 63
4.10 Binary decomposition tree for a series-parallel representation of the non-series-parallel graph from Fig. 4.7 63
4.11 DATE'19 MBIST benchmarks' structure 65
4.12 Robustness enhancement results, cost $\leq$ 10% 67
4.13 Robustness enhancement results, damage $\leq$ 10% 67
4.14 Tradeoff between the hardware cost and the damage, case-study 68

Chapter 5: Testable RSNs 70

5.1 Testability issues in the RSN example from Fig. 2.1 74
5.2 Scan cell with a DfT Enhancement (in green). The additional scan multiplexer allows to propagate the data from the shadow flip-flop to the scan flip-flop. 76
5.3 Testability concept example 79
5.4 Resynthesis example: The testability issue from Fig. 5.3 is resolved by inserting two scan cells 84
5.5 Test sequence construction a) workload sequence is augmented with a flush sequence b) bit sharing mechanism 85
5.6 RSN Online/Offline Self-Test Infrastructure (ROSTI) structure 87
5.7 Testability-enhancement for the RSN from Fig. 2.1 89
5.8 Periodic test 91
5.9 Possible configurations for Fig. 2.1 94
5.10 RSN chain: Each scan multiplexer is controlled independently 97
5.11 Active Scan Path in a Binary Decomposition Tree from Fig. 4.4 98
5.12 Added cells for enhancing the testability of the artificial benchmarks . . . 101
5.13 Fault coverage comparison 102
5.14 Test cost reduction by test integration 103
5.15 Test cost, number of activated paths (ASPs) 104
5.16 Test cost, number of sequences 105
5.17 Test cost reduction for 90% coverage compared to full coverage, cycles . . 105

Chapter 6: Security Compliant RSNs **108**

6.1 Security compliance violations in the RSN example 116
6.2 General flow of SCAR 118
6.3 Node cutting options 121
6.4 Graph construction 124
6.5 Secure RSN graph 125
6.6 Accessible RSN graph 127
6.7 Details of the filter-based protection 129
6.8 Limitations of filter-based schemes for the RSN from Fig. 2.1. Limiting the accessability to the instrument i_4 restricts the access to the instrument i_3. . 131
6.9 Limitations of resynthesis for the RSN from Fig. 2.1. Blocking the accessibility of the instrument i_4 only for certain user groups is not possible. . . . 132
6.10 Hybrid Protection scheme as a remedy. For the RSN from Fig. 2.1 the access specification is fulfilled for multiple user groups. 133
6.11 General flow of the hybrid protection scheme 136
6.12 Runtime decrease compared to the matrix-based computation 139
6.13 The reachability properties of the system and the RSN 140
6.14 The connectivities through RSNs 141
6.15 Relative number of changes per violation for the developed schemes 142
6.16 Filter applicability fraction of the initial RSNs 143

Chapter 7: Integration of Dependable RSNs **146**

7.1 Initial RSN 147
7.2 Structural changes to the RSN topology 148
7.3 Changes to the segments lengths 149
7.4 Changes to the properties of scan primitives 150
7.5 Dependable RSN 151

List of Tables

Table 5.1 Added cells for enhancing the testability of the original benchmarks 100

Table 7.1 Influence of dependability enhancing resynthesis schemes on RSNs 146

Table A.1 Characteristics of benchmarks 162

Table B.1 Robust RSN Synthesis, Initial Assessment 166
Table B.2 Robust RSN Synthesis, SPEA-II, Varying Optimization Criteria 167
Table B.3 Testability-oriented resynthesis to detect all UDT-PL 169
Table B.4 Design-for-Test Scheme Results: Test Cost 170
Table B.5 Design-for-Test Scheme Results: Overhead and Coverage 171
Table B.6 Periodic Test of RSNs: Genetic Algorithm 172
Table B.7 Periodic Test of RSNs: Sequence Optimization Results 173
Table B.8 Periodic Test of RSNs: Coverage 174
Table B.9 Security Compliance Analysis 175
Table B.10 Security Compliant RSN Integration 178
Table B.11 A Hybrid Protection Scheme: Overhead 179
Table B.12 A Hybrid Protection Scheme Comparison: Filter Applicability Fraction 179
Table B.13 A Hybrid Protection Scheme Comparison: Number of Structural Changes 180

List of Abbreviations

AVFS	Adaptive Voltage and Frequency Scaling
ABB	Adaptive Body Bias
ASIL	Automotive Safety Integrity Level
ATPG	Automatic Test Pattern Generation
BIST	Built-In Self-Test
CSU	Capture-Shift-Update
CNF	Conjunctive Normal Form
DNF	Disjunctive Normal Form
DT-PL	Detectable by an altered path length
DUT	Device under Test
EDA	Electronic Design Automation
ESC	Essential Select Condition
FAA	Filter Applicability Analysis
FSM	Finite State Machine
GA	Genetic Algorithm
HCI	Hot Carrier Injection
HDL	Hardware Description Language
ICL	Instrument Connectivity Language
ILP	Integer Linear Programming
(i)JTAG	(Internal) Joint Test Access Group
MOO	Multi-Objective Optimization
NBTI	Negative-Bias Temperature Instability
NFF	No-Failure-Found
NSGA	Non-dominated Sorting Genetic Algorithm
NP	Non-deterministic Polynomial-time

PDL	Procedure Description Language
PSV	Post-Silicon Validation
RSC	Relative Select Condition
RSN	Reconfigurable Scan Network
RTL	Register Transfer Level
ROSTI	RSN Online/Offline Self-Test Infrastructure
SAT	Boolean Satisiability
SCAR	Security Compliance Analysis and Resynthesis
SIB	Segment Insertion Bit
LSIB	Locking SIB
PSIB	Parallel SIB
S^2IB	Secure SIB
SLM	Silicon Lifecycle Management
SPEA	Strength Pareto Evolutionary Algorithm
TAP	Test Access Port
TAT	Test Application Time
TPG	Test Pattern Generation
UDT-PL	Undetectable by an (altered) path length
VLSI	Very Large Scale Integration
X-value	Unknown value

List of Symbols

	RSNs and instruments
DS	data scan segments
CS	control scan segments
M	scan multiplexers; m_j a scan multiplexer
$P := S \cup M$	scan primitives; p_j a scan primitive
SIG	control signals; sig_j a control signal
I	registers of instruments; i_j an instrument register
F	faults; f_j a fault
C	scan configurations, c_j a scan configuration
	Graphs
$G := (V, E)$	a graph with a vertex set V and an edge set E
$G^S := (V^S, E^S)$	a system graph
$G^I := (V^I, E^I)$	an instruments graph
$G^{RSN} := (V^{RSN}, E^{RSN})$	an RSN graph
	Connectivities between the vertices
$dp(v_j, G)$, $ds(v_j, G)$	direct predecessors and successors
$sp(v_j, G)$, $ss(v_j, G)$	structural predecessors and successors
$fp(v_j, G)$, $fs(v_j, G)$	functional predecessors and successors
$struct\_path(v_j, v_k, G)$	a structural path from the vertex v_j to the vertex v_k
$path(v_j, v_k, G)$	a functional path from the vertex v_j to the vertex v_k
$ESC(v_j, sig_1, ...sig_n)$	the essential select conditon of the vertex v_j
$RSC(v_j, v_k, sig_1, ...sig_n)$	the relative select conditon of the vertex v_j given that the vertex v_k is selected

Active Scan Paths and their properties	
$asps(G)$	active scan paths through a graph G
asp_k	an active scan path with an identifier k
$pathLen_k$	the length of asp_k
$conds(asp_k)$	the conditions for activating asp_k
$V(asp_k) \subset V^{RSN}$	a subset of vertices, which belong to the path asp_k
Test sequences	
$Seq_k := \{asp_0^k, asp_1^k, \dots asp_l^k\}$	a sequence of ASPs
$Test := \{Seq_0, \dots Seq_k, \dots Seq_n\}$	a set of test sequences
Violations and User-defined restrictions	
$viol(v_j, v_k)$	a security compliance violation from the vertex v_j to the vertex v_k
$Users$	user; u a user
$viol_u(v_j, v_k)$	a user-specific security compliance violation from the vertex v_j to the vertex v_k
$viol_u(v_j)$	an authorization violation for the vertex v_j
$G(u) \subset S$	a subset of scan segments, which are restricted for the user u
$A(u) \subset S$	a subset of scan segments, which are must remain accessible for the user u

Chapter 1

Introduction

1.1 System Dependability Management

Continuous efforts of the semiconductor industry to follow the Moore's law [Moo65, Moo75] still allow to increase complexity and higher integration in modern digital circuits. At the same time, to allow high-performance and low-power operation, these circuits are at the boundary of their operational limits [KEGT17]. Moreover, due to continuous technology scaling, the semiconductor manufacturing processes are increasingly more prone to defects and process variations [PBH$^+$11].

The complexity of modern systems makes it inevitable to consider such dependability aspects of system design and operation, as ensuring their reliability, availability, functional safety, and cyber-security. In Section 1.1.1, a short summary about the major dependability aspects is provided, followed by the classification of the dependability-enhancing means in Section 1.1.2. To manage the system dependability throughout the whole lifetime of the system, various validation, test, and diagnosis methods are applied. Various functional and non-functional dependability instruments are integrated into the system to support these methods, as described in Section 1.1.3.

1.1.1 Dependability Requirements of the Modern Devices

System dependability, as defined in [ALR04], is the ability to avoid system failures that are more frequent and more severe than acceptable. It serves as an integrating concept that unites such requirements to the system operation, as reliability, availability, maintainability, functional safety, confidentiality, integrity, and cyber-security. For more information about system dependability requirements, the reader is referred to [ALR04]. The remainder of this section provides more details about the specific dependability properties.

Reliability

Reliability is defined as the probability that a system would provide correct service longer than a given period of time t, or, in other words, as a readiness for correct service. Reliability $R(t)$ is defined with the following formula:

$$R(t) = Prob\{T > t\} = 1 - F(t) \tag{1.1}$$

where $F(t)$ defines the probability that a system is down at t.

Availability

Availability is defined as the system readiness to provide correct service at a given point of a time. Availability of a system can be calculated with the following formula:

$$Availability := \frac{Uptime}{Uptime + Downtime} \tag{1.2}$$

where $Uptime$ defines the time period, when the system provides correct service, while $Downtime$ is the time period when the system is out of operation. The latter time period includes planned and unplanned maintenance. Therefore, the term availability is usually used in those systems, where system recovery and/or repair is foreseen.

Maintainability

Maintainability is defined as the system ability to undergo maintainance, modifications, and repairs to ensure correct service. In order to increase maintenability of a system, a system must be designed in a way that faults may be detected, quickly isolated and - if possible, repaired.

Functional Safety

According to the international safety standard ISO 26262 "Road vehicles – Functional safety" [dN11], *functional safety* is defined as the “absence of unreasonable risk due to hazards caused by malfunctioning behavior of electrical/electronic systems”. In a functionally safe system, risks of dangerous failures caused by systematic and random faults, must be minimized. There exist certain risk classes, which are specific for the application field. For example, in automotive industry, Automotive Safety Integrity Levels (ASILs) are widely used. Based on the risk class, formal requirements to system design and operation are formalized and applied to ensure a sufficient level of safety.

Confidentiality

Confidentiality is defined as the ability of a system to operate in the absence of unauthorized disclosure of information. In a confidential system, sensitive information is only available for authorized entities and is kept away from other persons. This property can be ensured by using such security mechanisms, as user authorization and authentication, and also categorization of information into different levels of sensitivity, and encryption of sensitive data.

Integrity

Integrity is defined as the system ability to operate in the absence of improper system alterations by an unauthorized entity or due to non-human-caused events. In other words, the information transmitted through a system is available to the receiver in an intended format, and is not corrupted.

Security

Security [ALR04] of a system is defined as the concurrent existence of the following properties in the system:

(a) availability for authorized users only,

(b) confidentiality,

(c) integrity against unauthorized system alternations.

In modern systems, it is almost impossible to ensure functional safety without managing its security [VGY+16]. At the same time, to ensure dependable operation of systems, extensive test measures are applied, which in general increase the observability and the controllability of system's internals. This extra observability and controllability might serve as a possible side-channel for an attacker [RDN+14] and might be misused to compromise functional safety. An attack on car's breakage system is a remarkable example of an interplay between functional safety, security and test of a dependable system. For more details on system security refer to [TW11].

1.1.2 Dependability-Enhancing Means

To ensure the dependable operation of a system, an extensive number of different approaches to enhance system design and operation have been presented. The remainder of this section summarizes the most important classes of dependability-enhancing means.

Fault-Tolerance

Fault-tolerance (FT) methods [KK07] are those methods, which avoid service failures in the presence of faults by applying various types of redundancies. These redundancy types include structural, information, and time redundancies. In the case of structural redundancy, extra resources are incorporated to detect and possibly correct the effects of failed components. However, adding more resources to the system may reduce the system reliability [Lal00]. Some examples of structural redundancies include Triple Modular Redundancies (TMR) [LV62], hypercubes [Mil63], cube-connected cycles [PV79] and FT-trees [DH90]. Information redundancy exploits additional bits of information to detect and correct errors. Some important example are Error Correcting Codes (ECC) [Mac99, MM00] and Algorithm-Based Fault-Tolerance (ABFT) [KA84, BHW14]. Finally, time redundancy is based on re-execution, and is efficient, in particular, to detect and tolerate transient faults.

Fault-Avoidance

Fault-avoidance methods are applied to prevent the occurrence or introduction of faults into a system [MCXBA99]. Faults can be avoided by making all the components of the system more robust, e.g. through resizing [QM06, JRBS06]. There are various techniques available to reduce the probability of defects by hardening some components and cells locally. They are developed in the field of design-for-manufacturability [Dom12] and can reach as far as applying TMR locally to single cells [VBG+08]. They decrease the probability of defects and do not affect the test and diagnostic procedures but increase power consumption and area overhead. Hardening a minor number of carefully selected components, while leaving the remaining part of the device unprotected, still reduces the probability of a system failure and has acceptable costs [ZWPB08, PH11]. In the literature, hardening is usually applied for soft error mitigation [MZS+07, EKD+03, ORM07].

Fault-Removal

Fault-removal approaches reduce the number and severity of faults in a system. These approaches aim to detect the presence of faults by means of testing, locate them through diagnosis and consequently remove them [MCXBA99]. More details about major fault detection stages are provided in the next section.

Fault-Forecasting

Fault-forecasting methods estimate the present number, the future incidence, and the possible consequences of faults for system operation. Performing fault-removal and fault-forecasting allows deciding whether current specifications of system functionality and its dependability properties are adequate and if the system can fulfill them now and in the future.

1.1.3 Dependability Management throughout the Lifetime

To ensure the reliable operation of modern devices and keep the yield high, devices are tested throughout their whole lifetime. Large amounts of data are collected from the devices by using embedded instruments. The data collected in the earlier stages of the lifecycle can be reused later to optimize the latter stages. Moreover, the data obtained during the latter stages can enhance the design and manufacturing process of the next-generation devices.

In this section, we provide an overview of the integration and usage of embedded instruments throughout different stages of silicon lifecycle management (SLM). These stages include the *design*, *post-silicon validation*, *manufacturing test and diagnosis* and *in-field* stages. The fifth *field-return* stage extends the SLM concept even more by considering the data obtained after the system's end of life. It helps address the returns, in particular the No-Failure-Founds. For more information about testing and diagnosis, refer to [ABF90, WWW06]. The remainder of this section provides some details about the necessary steps to integrate and operate embedded instruments.

Design Stage

Modern semiconductor devices integrate a vast amount of instruments. Dependability instruments include, e.g., sensors, monitors [SKKN20, LSW20], built-in self-test (BIST) registers [RTKM04, HWZ$^+$07, KMM$^+$20], Adaptive Voltage and Frequency Scaling (AVFS) blocks, and Adaptive Body Bias (ABB) control units [TKD$^+$07], temperature control and error rate adoption units.

Post-Silicon Validation

In earlier technologies, a large portion of hardware mismatches could be detected during the pre-silicon phase of the design lifetime. Due to the physical limits and also the system complexity of modern devices, it is no anymore possible to rely only on pre-silicon design verification to detect all design mismatches. Post-silicon validation is a test insertion that is applied on a minor number of manufactured device prototypes. It validates the compliance of the actual implemented device behavior over the initial specification. The prototypes are thoroughly examined, and possible faults after production are detected [MSN10]. Error localization during post-silicon validation is supported by an extensive number of on-chip sensors and trace buffers. Based on the results of this phase, the design is enhanced in the next generations by patching, circuit editing, or re-spinning the device. This whole process is repeated until no further problems are found.

Manufacturing Test and Diagnosis

After design errors are detected and resolved during pre-silicon verification and post-silicon validation, volume production of the device starts. Due to the complexity of modern devices and technology node scaling, manufacturing defects are inevitable during volume production. To precisely detect the defects and thereby ensure high reliability of devices, while preserving high yield, offline manufacturing test and diagnosis methods are applied. Numerous instruments, such as Built-In Self-Test blocks, are used to support the test. Additionally, analyzing test and diagnostic results helps to find weak spots in the manufacturing process and consequently improve it.

In-Field Operation

After devices are manufactured and tested, they are shipped to the customer, and online reliability management is used to observe and possibly control the device degradation [IK19]. The degradation arises due to extreme environmental conditions and also due to aging mechanisms, which include but are not limited to Negative Bias Temperature Instability (NBTI) [APZM07] or Hot Carrier Injection (HCI) [MSVK06]. System in-field operation can be supported by embedded instruments, which include e.g. monitors and sensors [LSW20]. In the case of performance degradation, fault handling mechanisms are used to guide the system reconfiguration or to adjust the operating conditions. Thereby, the useful lifetime of a system can be prolonged.

Field Return Diagnosis

At the end of the lifetime, diagnostic data from the system can be further analyzed to identify possible root causes of failures, in particular those which are marked as no-failure-found (NFF). This data is collected by using an extensive number of instruments throughout the whole life of the system and can be used to optimize the manufacturing process and even the design of next-generation systems.

1.2 Reconfigurable Scan Networks (RSNs) for Dependability Enhancement

Reconfigurable Scan Networks (RSNs) provide flexible and scalable access to an extensive number of extra-functional embedded instruments. They are standardized by IEEE Std. 1149.1 [IEE13] and IEEE Std. 1687 [IEE14] and are also commonly referred to as iJTAG (internal Joint Test Access Group) networks or IEEE 1687 networks. RSNs are used throughout the system lifetime and thereby support the dependability management, as described in Section 1.2.1. At the same time, improper RSN integration can affect the system dependability, as discussed in Section 1.2.2. A high-level overview of the developed automated framework for integration of dependable RSNs is given in Section 1.3.

1.2.1 Dependability Management Support via RSNs

Reconfigurable Scan Networks access the non-functional instruments through scan paths, which all start at a scan-in port and end at a scan-out port. In Fig. 1.1, the instruments shown in orange, such as sensors and BIST registers, are accessed via an RSN. Instruments are accessed for observation and control via an activated scan path (ASP), which allows to dynamically include those instruments, which must be accessed now. The values in the control registers $\{cs_1 \ldots cs_5\}$, shown in yellow, determine the currently configured scan path. Vertical arrows show possible points to select different paths. As shown with a dashed line in Fig. 1.1, an initial active scan path traverses the control register cs_4, as well as the segments s_3, s_4 which access the BIST register and the aging monitor.

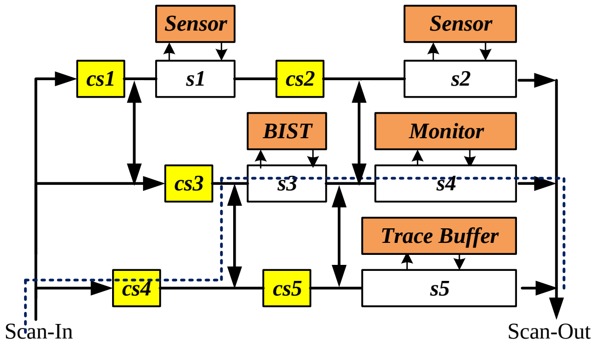


Figure 1.1: RSN accessing the instruments

Initially, RSNs have been presented to support the offline stages of the lifecycle, which include *post-silicon validation* and *manufacturing test*. During post-silicon validation, RSNs collect as much data as possible from a limited number of prototypes, and this data is further used to detect and locate faults. During the manufacturing test, RSNs are used to provide test patterns and to fetch test data from the instruments. Analyzing the responses of RSNs allows detecting manufacturing defects.

The benefits of using RSNs to support Silicon Lifecycle Management are not limited to offline operation. Instead, nowadays RSNs are also increasingly used *in-field*. Some of the important online applications of RSNs are listed below:

- The in-field operation of the instruments can be controlled by RSNs. E.g. runtime-adaptive instruments, such as adaptive voltage and frequency scaling, error rate adoption, or temperature control, can be accessed by RSNs, or RSNs can be used to control online BISTs [RTB^{+}19].

- Reliability threats, such as faults or abnormal changes in the system behavior [IK19, ZNL16] can be fetched by RSNs and detected.

- The data about failures in certain modules [TJSD18] can be collected by RSNs.

- In the case of performance degradation or failure of specific modules, RSN-based fault-handling mechanisms can be used to guide the reconfiguration of a system or to adjust the operating conditions by controlling runtime-adaptive instruments through RSNs and thereby prolong the useful lifetime of a system [SDJ16].

- Security attacks can be detected with a help of RSNs [EKC18, RBT18].

1.2.2 Dependability Issues due to Improper RSN Integration

The dependability of the underlying system can be affected through an improper RSN integration. The remainder of this section provides an overview of the issues with regard to the most relevant dependability aspects, such as robustness, testability and security compliance of RSNs.

Robustness

Since RSNs occupy a significant fraction of the chip area, the probability of a fault therein is not negligible. Even a single fault in the RSN could corrupt scan paths, an erroneous data may be fetched by the RSN, make the instruments inaccessible through the RSN, and eventually result in a system failure. During the post-silicon validation (PSV) stage, an innovative process or a new design may result in an increased defect rate. Therefore, a single fault in an RSN may prevent accessing a significant part of instruments. As a result, a large portion of the validation data cannot be extracted from the device. Later, during the runtime stage, the system operation may be guided by runtime-adaptive instruments. Inaccessibility of such critical instruments due to a single fault in the RSN may cause a system failure or even permanent damage.

Testability

Specific fault effects in RSNs are observable only for certain RSN configurations, and sequential test pattern generation for such faults is unfeasible for large networks. The known RSN test solutions, presented in [CMR$^+$15, KBSW16, CZP$^+$18, HHD21], e.g., may not detect all the faults due to inaccessibility of certain RSN components, and do not provide complete testability enhancing solution. An undetectable fault in an RSN may cause silent data corruption if data is captured from or updated to the wrong instrument.

Moreover, conventional methods [UKW17, KBSW16, CDRS20, CSSS18, HHD20] test RSNs once, after the manufacturing, or perform online concurrent test of RSNs [WLA$^+$21]. To avoid fault accumulation concurrent test has to be complemented by a periodic test, since rarely used components may be still subject to aging [LSW20]. The need for the periodic test is especially strong in safety-critical applications like automotive [KMM$^+$21, EZ17].

Security Compliance

To ensure system-level security, a system designer thoroughly develops the connections inside a system in a way that prevents unauthorized access and information leakage. A design-for-test (DfT) integrator might not be fully aware of all the designer's security intentions and might integrate the RSN in a way, which is not compliant with the initial security properties. The additional connectivities in the system, introduced due to improper RSN integration, might be exploited by an attacker as a side-channel to leak or manipulate sensitive data or alternate the system behavior [EKC21, KDD16, RBT18, KBW17]. It must be ensured that the original security analysis and protection policy are not invalidated by the DfT-infrastructure.

Combination of the Dependability Aspects

The above-mentioned dependability aspects might be contradicting in general. In particular, the increased testability of a system may open a side-channel for an attacker [RDN$^+$14]. The dependability aspects must be considered altogether and all the contradictions between them must be mitigated at the design stage.

1.3 Overview of the Work

The document at hand establishes a methodology to integrate dependable RSNs for a given system. The developed methodology considers the most relevant dependability aspects, such as robustness, testability, and security compliance of RSNs. It overcomes specific challenges, which arise at different stages of the Silicon Lifecycle, and is scalable with the increasing size and complexity of RSNs. The remainder of this section provides an overview of the contributions and explains, how the document is organized.

1.3.1 Major Contributions

Fig. 1.2 presents the major aspects of dependability-aware RSN integration:

- *Dependability aspects:* To support an efficient system dependability management, RSNs themselves must remain dependable. A dependable RSN is characterized by such dependability aspects, as a possibility to provide robust access to the instruments for observation and control, testability of an RSN itself, as well security compliance with the intentions of the system designer. The presented scheme integrates dependable RSNs and mitigates the contradiction between enhancing the above-

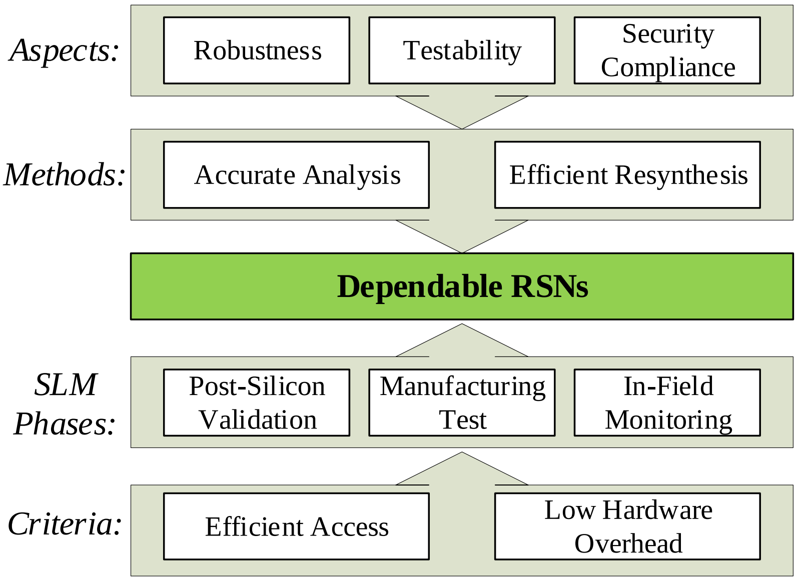


Figure 1.2: Objectives and Contributions

mentioned dependability aspects of an RSN. Thereby, it allows for the integration of a dependable RSN, which is suitable for the system requirements.

- *Methods:* Compared to conventional scan chains, analyzing the dependability aspects of RSNs and, in particular, enhancing those is even more challenging due to the high sequential depth of RSNs, the distributed control structure, as well as the complex sequential and combinational dependencies [BKW15b]. The first major contribution of this thesis is an exact and scalable analysis methodology for the dependability aspects above. This analysis serves as a basis for an efficient dependability-enhancing resynthesis method, which is the second major contribution.

- *Silicon Lifecycle Management (SLM) phases:* The presented methodology considers the integration of RSNs, which provide efficient instrument access throughout the whole silicon lifecycle. It covers the whole timespan starting from the post-silicon validation phase of the prototypes, through the manufacturing test and diagnosis during volume production, and finally supports the online dependability management and in-field monitoring.

- *Criteria:* The structure of RSNs is optimized to comply with the requirements of the design-for-test engineer, and is compatible for online and offline operation. To keep the test costs low, it is extremely important to reduce the RSN hardware overhead. Throughout the silicon lifecycle, especially during the manufacturing test and the online operation optimizing the test access time becomes inevitable.

1.3.2 Document Organization

The **first part** has provided a motivation, why the developed methodology to integrate dependable RSNs is essential to support the system's dependability. The remainder of this chapter presents an overview of how the document at hand is organized.

The **second part** includes the theoretical background, which is required for understanding the developed methodology. The necessary background on Reconfigurable Scan Networks is presented, including the information about the used fault models in RSNs. Next, the details are presented about the graph-based modeling of RSNs.

The state-of-the-art solutions for enhancing specific dependability properties of RSNs are provided in the **third part**. The major applications of RSNs are summarized in **Chapter 3.1**. In **Chapter 3.2**, the state-of-the-art solutions to provide reliable access via RSNs are presented. **Chapter 3.3** discusses the existing test and diagnosis methods for RSNs. **Chapter 3.4** discusses the existing ways to analyze and enhance RSNs for the security. Finally, **Chapter 3.5** summarizes the limitations of the existing methods and highlights the necessity of integrating dependable RSNs despite the presented research challenges.

The **fourth part** presents the developed methodology for integrating dependable RSNs. The major contributions are discussed in **Chapters 4 – 7**:

Chapter 4 ("*Robust RSNs*") provides the first solution to significantly enhance the robustness of RSNs by the means of cost-efficient hardening. The presented solution has been first published in [LWW22c]. The goal of the developed scheme is to ensure reliable access to the instruments for control and observation via RSNs throughout the lifecycle. During the post-silicon validation, the number of device prototypes is limited and the error rate is rather high. Using the developed method, the validation data can be extracted, since the major parts of the instruments remain accessible. During the runtime application, robust RSNs support reliable access to runtime-adaptive instruments.

In this chapter, a precise criticality analysis method is presented to assess the influence of faults in scan primitives on the accessibility of the instruments. Based on the analysis results, an efficient resynthesis method substitutes a minimized number of RSN primitives with high-yield cells to ensure the desired accessibility of the instruments via RSNs in the presence of defects.

Chapter 5 ("*Testable RSNs*") presents the first complete design-for-test (DfT) approach for RSNs, which allows generating test sequences with complete fault coverage for the usual fault models and has been published in [LWW21, LWW22a]. First, fault locations are identified, which cannot be tested by using the existing test methods due to the low observability and controllability of specific RSN parts.

The developed DfT methods ensure fault detection for three major parts of a Reconfigurable Scan Network: scan interfaces, scan segments, and control lines. The complete DfT scheme precisely identifies untestable faults in a given RSN and efficiently resynthesizes the RSN in a way that all the faults are testable by using the existing methods. An efficient test integration scheme is developed, which dramatically decreases the overall test access time compared to testing the individual parts of an RSN independently. Each test sequence contains a workload sequence, which is used to test faults affecting scan interfaces and control lines, and a short self-generated pre-sequence to concurrently test the shift logic of the segments on the activated path. The presented scheme is flexible with respect to the fault model, has a low hardware overhead, and does not require changing the RSN topology rules.

Testing the shift logic of the scan segments concurrently to the functional access, as shown in the first part of the chapter, is not enough to guarantee reliable access through RSNs. To avoid fault accumulation, the concurrent test has to be complemented by a periodic test, since rarely used components may be still subject to aging [LSW20]. The second part of the chapter discusses the first online periodic test method for RSNs, which has been first presented in [LWW22b].

The developed algorithm generates a short sequence of test patterns, which tests all scan primitives of an RSN. The overall test application time is minimized to comply with the timing requirements of the well-known safety standards. The generated sequences can be uploaded on-chip and applied periodically online to avoid fault accumulation in RSNs. Alternatively, the generated test sequence set can be reused throughout the system lifetime, e.g., for structural test and post-silicon validation of RSNs.

Chapter 6 ("*Security Compliant RSNs*") presents a complete approach for security preserving integration of RSNs. The developed approach ensures the compliance of the resulting RSN with the requirements of the original system.

The first section presents the first security compliance analysis approach from [LAR+19, LWW22d]. The developed approach overcomes the high sequential depth of RSNs and accurately analyzes the RSN functional dependencies considering retargeting. The information about the functional connectivities through RSNs and the connectivities between the instruments in the initial system are combined. The so-called hybrid connectivities between the instruments are identified for the case, when a given RSN is integrated into the system. These hybrid connectivities are compared to the initial connectivities between the instruments, and unwanted extra-connectivities due to improper RSN integration are identified as security violations.

In the second part of the chapter, a purely structural resynthesis scheme from [LAWW20, LWW22d] is presented. Based on the results of a security compliance analysis, the developed resynthesis approach resolves all the identified violations by applying a minimized number of structural changes to the RSN structure. An efficient divide-and-conquer heuristic is presented, which avoids exponential complexity in the average case. The automated resynthesis is flexible and allows to specify additional optimization criteria based on the DfT integrator's needs.

The third part of the chapter presents a flexible protection scheme for multiple user groups with different access permissions, as published in [LWW21]. It combines the benefits of the structural method presented in the second part of the chapter and the filter-based protection from [AKS+18]. The number of structural changes is dramatically reduced compared to the purely structural solution above, since only those violations are resolved structurally which cannot be handled by using sequence filters without sacrificing the accessibility of the instruments. The remaining violations are resolved functionally by using filters. The resulting RSNs are compliant with the security specification and do not extend the allowed connectivities between the instruments of the initial system.

Chapter 7 ("*Dependable RSNs*) presents a complete approach for integration of dependable RSNs. It discusses possible contradictions between specific dependability properties of RSNs and presents a flow to efficiently combine these properties in one dependable RSN.

Finally, **Chapter 8** provides a conclusion and serves as an overview of future research directions.

The appendices present the data for an evaluation of the developed methods on a broad range of widely-accepted benchmarks. Here, **Appendix A** presents the experimental setup, which is common for all the experiments, while **Appendix B** provides the details about the individual experiments for the contributions above.

1.4 Summary

To support the design and integration of modern systems, it is inevitable to consider such dependability aspects as reliability, availability, functional safety, and cyber-security. An increasing number of dependability instruments are integrated to manage the system's dependability throughout its lifecycle.

Reconfigurable Scan Networks provide flexible and efficient access to the instruments. However, their integration may result in dependability challenges. Faults in RSNs may affect the accessibility of the instruments through the RSNs. If these faults remain undetected, they may cause Silent Data Corruption. Additional unwanted connectivities may be introduced into the system due to improper RSN integration. They may be misused by attackers to leak some sensitive data or to manipulate the underlying system. All the dependability challenges above might lead to a system failure, permanent damage, or a functional safety threat if safety-critical components of the system are affected. Such properties of RSNs as high sequential depth and complex control dependencies make the analysis and enhancement of RSNs extremely challenging.

In this document, an automated framework is presented to precisely analyze and efficiently enhance such dependability properties of RSNs, as accessibility, robustness, and security compliance. The developed framework is scalable for large and complex RSN designs. The resulting RSNs are applicable throughout the whole system's lifetime. The remainder of the document discusses the individual contributions in the context of the existing state-of-the-art solutions. The last chapter discusses a complete solution to integrate dependable RSNs into the original system, such that the developed methods are not contradicting each other.

Part I

Background and State of the Art

Chapter 2

Reconfigurable Scan Networks

With the first efforts starting at the beginning of the century [REP+05], RSNs have been finally standardized in IEEE Std. 1149.1 [IEE13] and IEEE Std. 1687 [IEE14]. RSNs are integrated during the *design* stage of dependability management and access the embedded instruments throughout the whole system lifetime. Section 2.1 provides a short description of an RSN structure, followed by the specifics of accessing the embedded instruments through RSNs in Section 2.2. In Section 2.3, possible fault locations and their effects on the RSN functionality are introduced. Section 2.4 discusses the modeling of RSNs, as well as modeling of interfaces between RSNs and the functional system.

2.1 RSN Structure

A small part of a considered system is shown in Fig. 2.1. As already mentioned in Section 1.2.1, the considered system comprises the non-functional instruments' registers I (in the upper part of Fig. 2.1) and the RSN, which accesses the instruments' registers through the scan segments S (in the lower part). This RSN is used as a running example for the remainder of the document.

RSNs are constructed out of basic building blocks, which are commonly referred to as the *scan primitives*. The set of scan primitives P includes all the scan segments S and all the control primitives M. *Scan segments* are the scan primitives, which access the instruments through a parallel interface and which are used to shift the data through the RSN.

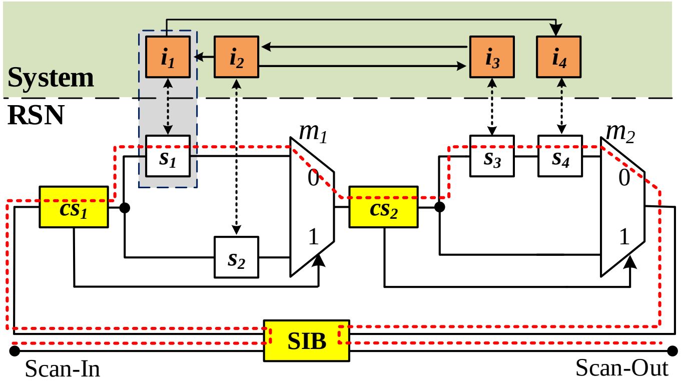


Figure 2.1: Running RSN example

Each scan segment contains a shift register and an optional shadow register, as shown in Fig. 2.2. The following types of scan segments can be distinguished:

- *Data scan segments* $ds \in DS$ are scan segments, where the shadow registers serve as an intermediate storage for the information, which is sent to the instruments.
- *Control scan segments* $cs \in CS$ are scan segments, where the information stored in the shadow registers is used to drive the internal control signals.

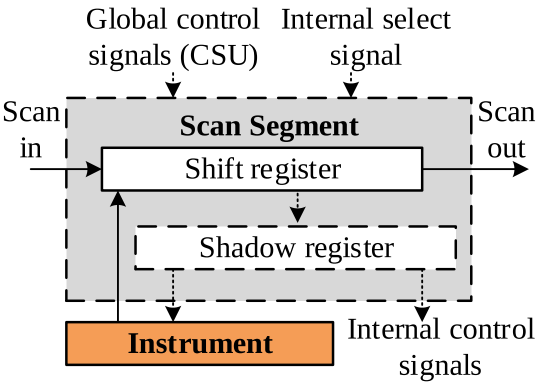


Figure 2.2: Scan segment internals

Control primitives are the scan primitives, which define the configured active scan path depending on the values of the *control signals*. If a control signal sig comes from outside of the RSN, it is referred to as an external control signal. If a signal comes from a shadow register of a control scan segment, it is called an internal control signal. The following control primitives are commonly used to build RSNs:

- *Scan multiplexers* select between appropriate parts of an RSN depending on the value of an address control signal and include them into an activated path.
- *Segment Insertion Bits (SIBs)* include or exclude specific parts of an RSN from an activated path depending on the control signal assignments.

Other types of control primitives might include multi-input scan multiplexers, and also different types of SIBs. To stay general, we assume that any control scan primitive is constructed as a combination of one or multiple scan multiplexer and an optional scan segment.

Example *Each SIB (as shown in Fig. 2.3.a) can be represented as a combination of a scan segment and a scan multiplexer, as shown in Fig. 2.3.b. The underlying segment is only selected, if the SIB is asserted. If the SIB is deasserted, the segment is bypassed.*

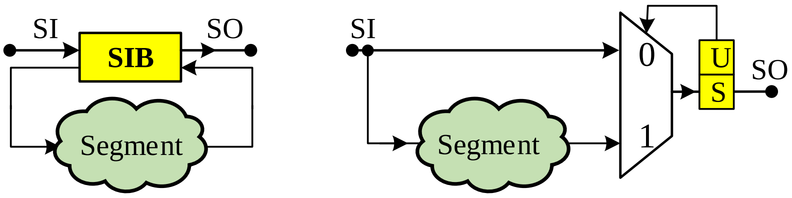


Figure 2.3: SIB transformation

There exist some types of RSN structures, which restrict possible connectivities within RSNs to specific regular structures. A widely-used example of such special-type RSNs is the so-called *SIB trees*. In SIB trees, only SIBs are used as configuration scan primitives. SIB trees are constructed recursively, such that each SIB provides access to one or multiple other SIBs or scan segments.

2.2 RSN Operation

At each particular time point, a *Scan Configuration* c is defined by the state of the scan primitives. An *Active Scan Path* (ASP) asp is an acyclic path through selected scan primitives from a primary scan-in to a primary scan-out. The scan primitives are explicitly selected by setting the *internal select signal* to logic one. In Fig. 2.1, an initial path starts at the primary scan-in port (SI), goes towards the SIB, traverses the segments cs_1, s_1, cs_2, s_3, returns back to the SIB and ends at the primary scan-out (SO).

In a valid scan configuration, only one active scan path can be activated at a time, and only those scan primitives, which are included into the ASP, can be used to access the corresponding instruments.

In the initial standard [IEE14], an RSN was supposed to be accessed via a JTAG Test Access Port (TAP), while the recent standard proposal IEEE P1687.1 [CVTR20] allows to use of functional ports to access and control RSNs. In [BKW15b] a temporal abstraction represents each access to an RSN as an atomic Capture-Shift-Update (CSU) operation. A CSU operation allows performing a transition from one scan configuration to another. During the *capture*-phase, the data from the instruments are read into the selected shift registers. During the *shift*-phase, the data is shifted towards the scan-out and the new data is being shifted in. During the *update*-phase, the data is latched in the shadow registers. This data can be written to the instruments' registers or used to trigger the control signals. The access phases mentioned above are controlled by the $CaptureEn$, $ShiftEn$, and $UpdateEn$ signals. These signals are shown as *global control signals (CSU)* in Fig. 2.2.

Multiple CSU-operations may be needed to transport the data to a certain target or to include necessary scan segments into an ASP. A *Transition Relation* defines the pairs of scan configurations (c_1, c_2), such that a transition from c_1 to c_2 requires only one CSU-operation. Computing the control patterns to perform these CSU-operations is called *retargeting*. For more details on CSU-modeling, see [BKW15b].

Each scan segment (Fig. 2.2) consists of one or multiple scan cells. A gate-level structure of a single scan cell is shown in Fig. 2.4. It consists of a scan flip-flop and an optional shadow flip-flop, which are connected as follows:

- *A shift path* starts at the scan-input (SI) and ends at the scan-output (SO) of a cell, and contains two multiplexers ($M1$ and $M2$) and an internal data path of the scan flip-flop. During the shift phase of a CSU operation, the multiplexer $M1$ propagates the data from the scan-input through the multiplexer $M2$ and the scan flip-flop towards the scan-output.

- *An update path* starts at the scan flip-flop's output and leads to the data output Q, which may be connected to an instrument or may drive RSN-internal control signals. The path comprises the multiplexer $M3$ and the internal data path of the shadow

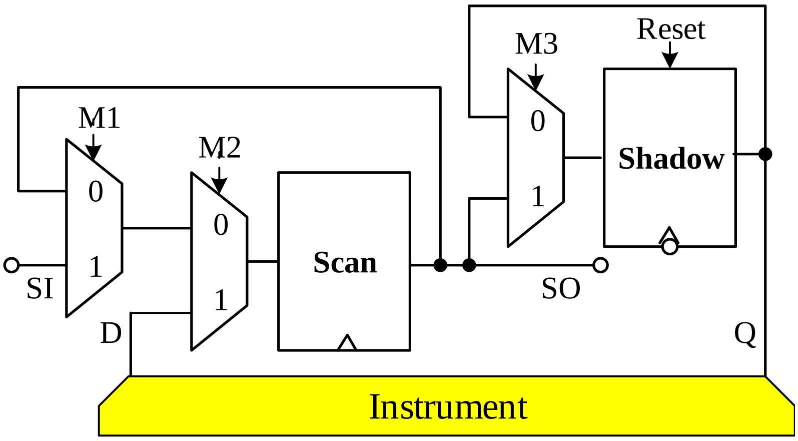


Figure 2.4: Scan cell accesses an instrument

flip-flop. During the update phase, the multiplexer $M3$ propagates the data from the shadow flip-flop to the output Q.

- *A capture path* starts at the data input D, traverses the multiplexer $M2$ and the data path of the scan flip-flop. During the capture phase, the multiplexer $M2$ propagates the data from the instrument into the scan flip-flop.

2.3 Faults in RSNs

A fault in an RSN may affect the interfaces to the instruments, the control signals or the scan segments. The remainder of this subsection discusses possible fault locations and their effects on the RSN functionality.

2.3.1 Faults at Interfaces to Instruments

The communication with the attached instruments and generation of internal control signals can be affected by faults in the shadow flip-flops, as well as at the multiplexer $M2$ and $M3$ in Fig. 2.5, as shown with red color and explained below:

- *Multiplexer $M3$ and data path of a shadow flip-flop:* A timing violation affecting the shadow flip-flop or a fault at the multiplexer $M3$ may corrupt writing the data to the instrument during the update phase.

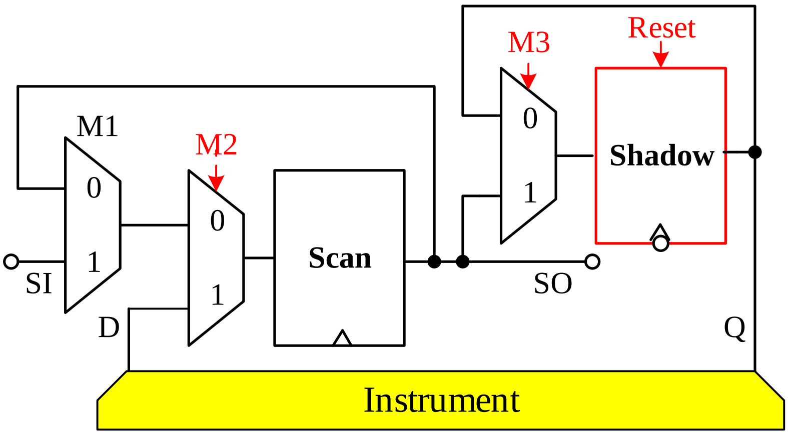


Figure 2.5: A scan cell with injected faults

- *Multiplexer $M2$:* If the multiplexer $M2$ is faulty, it may prevent from reading correct data from the instrument during the capture phase.

- *Reset line of a shadow flip-flop:* If the reset line of a shadow register is affected by a stuck-at-0 fault, it may not be possible to reset its value to an initial known state.

The logic around a shadow register can only be set and observed via the instrument. In case of an analog instrument, such as a digital to analog converter (DAC), the observability of the data is low, since the information is masked while being propagated through the instrument. This makes the faults at the multiplexers $M2$ and $M3$, as well as the faults affecting the shadow flip-flops, in general not testable.

Example: *In Fig. 2.1, a fault might affect an interface between the scan segment s_1 and the instrument i_1, as shown with a grey box. If the capture-circuitry of the scan interface of the scan segment s_1 is faulty, incorrect data can be latched into the instrument i_1.*

2.3.2 Faults in Control Primitives

Faults in the control primitives such as the scan multiplexers and the SIBs may arise due to defects in control lines or internal defects in the control primitives. These faults are usually modeled as high-level "stuck-at" faults, as defined in [CZP$^+$18]:

- *Scan Multiplexers*: If a scan multiplexer always selects a specific input with an identifier id, regardless of the assignment to the address control line, we say that this scan multiplexer is affected by a "*stuck-at-id*" fault.
- *SIBs*: If a SIB always provides access to the underlying segment, regardless of the applied access pattern, we say that this SIB is "*stuck-at-asserted*". If access to the underlying segment is never provided, the SIB is "*stuck-at-deasserted*".

Example: *Assume a scan multiplexer m_1 from Fig. 2.1 is affected by a stuck-at-1 fault. Due to this fault, the scan segment s_1 becomes inaccessible, which leads to the inaccessibility of the instrument i_1 via the RSN.*

2.3.3 Faults in Scan Segments

Examples of faults, which affect the primitives located on the shift path of a scan segment, include setup- and hold-time violations in the corresponding scan flip-flops. These violations might prevent correct data from being latched into the flip-flops while shifting.

Example: *If the scan flip-flop of the scan segment s_2 from Fig. 2.1 has a setup-time violation, the data is not properly latched in this flip-flop. The propagation through the activated path which traverses the scan segment s_2 is affected.*

2.4 Graph-Based Modeling of RSNs

Graphs are frequently used to represent the connectivities between logic gates, dominance relations, and other structural and functional properties of digital circuits. Thereby, in this document, a graph-based model of RSNs is used as a basis for the developed analysis and resynthesis approaches.

The remainder of the section is organized as follows. The first subsection provides some basic definitions of graphs and, in particular, trees. The second subsection discusses graph-based modeling of RSNs and their connectivities to the instruments of the original system.

2.4.1 Basic Definitions

Definition 2.1 (Graph). A *graph* G is a pair (V, E), where V is a set of vertices and E is a set of edges (paired vertices).

Definition 2.2 (Directed graph). A *directed graph* $G := (V, E)$ is a graph, where an edge set E, which contains ordered pairs of vertices, is defined as follows:

$$E \subseteq \{(v_i, v_j) | (v_i, v_j) \in V^2; i \neq j\} \tag{2.1}$$

A cycle is a graph, where vertices V can be ordered $(v_1, \ldots v_n)$ such that the edges are $\{v_i, v_{i+1}\}$; $i \in (1, n-1)$ plus the edge $\{v_n, v_1\}$. A graph is referred to as an acyclic graph, if it does not contain any cycle. Otherwise, a graph is referred to as a cyclic graph.

Vertices and edges of a graph can be annotated with some additional information, e.g. with an integer number or a Boolean formula [Boo12]. If a certain vertex does not have any incoming edges it is referred to as a source of the graph, while a vertex with zero outgoing edges is called a sink of the graph.

Definition 2.3 (Tree). A *tree* is an undirected graph, in which any two vertices are connected by exactly one path.

Tree decompositions of graphs are frequently used to solve specific computational problems in a scalable and efficient way, e.g. by exploiting divide-and-conquer paradigm or dynamic programming. Depending on the needs of a specific algorithm, there exist the following standard notations to traverse trees, as shown for the example in Fig. 2.6:

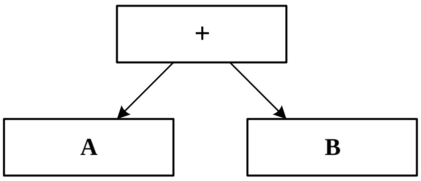


Figure 2.6: Graph traversal notations

- *Prefix notation* or Polish notation is a mathematical notation, where the *root* node (the operator) is traversed before all its children (the operands). In Fig. 2.6, the nodes would be processed following the order: $+AB$.

- *Infix notation* or in-order is a notation, where first the left child is processed, then the root vertex, and finally the right child. In Fig. 2.6, the in-order is $A + B$.
- *Postfix notation* or reverse Polish notation is a notation, where the *root* node is processed *after all its children*. In Fig. 2.6, the order would be $AB+$.

Using the notations above for the whole tree, efficient implementations of depth-first-search, breadth-first-search and other routines are developed. For more information about graphs, refer to [Big74].

2.4.2 RSN Graph

The whole considered system is modeled as a directed graph $G^S := (V^S, E^S)$ with vertices V^S and edges E^S. A *system graph* for the example RSN from Fig. 2.1 is shown in Fig. 2.7.

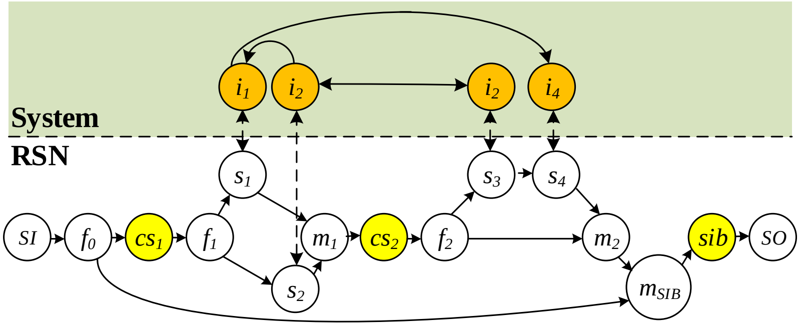


Figure 2.7: Example graph of an RSN from Fig. 2.1

It is constructed of two subgraphs, namely the instruments graph $G^I := (V^I, E^I)$, the RSN graph $G^{RSN} := (V^{RSN}, E^{RSN})$, and the edge set E^{CON}, which represents the connections between the subgraphs. The system graph is defined as follows:

$$V^S := V^I \cup V^{RSN} \tag{2.2}$$

$$E^S := E^I \cup E^{RSN} \cup E^{CON} \tag{2.3}$$

The instrument graph is constructed as follows. The vertex set V^I contains all the registers of extra-functional instruments, which are accessed through the RSN. The edges E^I

represent the direct connectivities between the vertices of the graph or the connectivities through the functional registers or the combinational blocks.

An RSN is modeled as a directed graph $G^{RSN} := (V^{RSN}, E^{RSN})$, where V^{RSN} is the vertex set and E^{RSN} is the edge set. Each vertex corresponds to a scan primitive, a primary scan input or output, or represents a fan-out stem, as shown in Fig. 2.7 for the example from Fig. 2.1. We assume a single scan input $SI \in V^{RSN}$, and a single scan output $SO \in V^{RSN}$. Each edge models a direct connectivity between the vertices.

The set $ds(v_j, G)$ of *direct successors* for a vertex v_j contains the nodes reachable by a single edge $\{v|(v_j, v) \in E\}$. The *direct predecessors* $ds(v_j, G)$ are defined similarly.

The vertex m_j is structurally reachable from the vertex m_i, if at least one path exists from m_i to m_j. The set $ss(v_j, G)$ of *structural successors* for a vertex v_j contains the structurally reachable nodes. The *structural predecessors* $ss(v_j, G)$ are defined similarly.

Definition 2.4 (Structural Path). A *Structural Path* $struct\_path(v_j, v_k, G)$ between the vertices v_j and v_k is a sequence of vertices $v_j \dots v_i, v_{i+1} \dots v_k$, where $v_{i+1} \in ds(v_i, G)$.

Definition 2.5 (Functional Path). A *Functional Path* $path(v_j, v_k, G)$ between the vertices v_j and v_k is a structural path, where a valid assignment to the logic signals exists, which selects all the vertices on the path simultaneously.

A *Hybrid Functional Path* is a functional path, which might traverse any combination of subgraphs in the system graph G^S. In Fig. 2.7, a hybrid path from the scan segment s_4 to the instrument i_7 can be represented as follows:

$$path(cs_1, i_3, G^S) := path(cs_1, s_2, G^{RSN}) \& path(s_2, i_2, G^S) \& path(i_2, i_3, G^I) \tag{2.4}$$

The set of functional successors $fs(v_j, G)$ of a vertex v_j contains the structural successors v, such that there exist a functional path from v_j to v, or the data from the vertex v_j can be transmitted to the vertex v by activating multiple functional paths. The functional predecessors $fp(v_j, G)$ are defined in a similar way.

An *active scan path* (ASP) $asp_j := \{v_0, \dots, v_i, v_{i+1}, v_k\}$ is a functional path in the RSN graph G^{RSN}. For an ASP asp_j, the vertex set $V(asp_j) \subset V$ includes the vertices which belong to this path. The initial state $c_0 \in C$ corresponds to the initial active scan path asp_0, and includes the vertices $V(asp_0) \subset V$.

An access to an instrument of an RSN might require the sequential activation of multiple scan paths. A sequence of paths $Seq_k := \{asp_0^k, asp_1^k, \ldots asp_i^k\}$ starts from the initial state asp_0, which is obtained by applying a global reset.

The following relations are determined:

- *Reconvergence vertex [MR90]:* The vertex m_j is a reconvergence vertex of the vertex m_i, if there are two paths $path_1$, $path_2$ with the corresponding vertex sets $V(path_1)$ and $V(path_2)$ such that $V(path_1) \cap V(path_2) = \{m_i, m_j\}$, m_i is the source of both $path_1$ and $path_2$, and m_j is their sink.
- *A closing reconvergence* of a vertex m_i is such a reconvergence vertex m_j, which does not reach any other reconvergence vertex of the vertex m_i.
- *The reconvergency region* of a vertex m_i includes all the vertices, which are reachable from this vertex and also reach its closing reconvergence.
- For two vertices m_i and m_j, we say that the vertex m_j *dominates* the vertex m_i, if all the paths through the vertex m_i to SO also traverse the vertex m_j.
- A *child* of a vertex m_j is a vertex m_i, which belongs to a reconvergency region, where the vertex m_j is a closing reconvergence. The vertex m_j is referred to as a *parent* of the vertex m_i.
- A *neighbor* of a vertex m_i is a vertex m_j, which belongs to a reconvergency region, which does not intersect with the reconvergency region of m_i, such that either both vertices m_i and m_j either have a common parent vertex m_k or they do not have any parent vertex.

2.5 Summary

Reconfigurable Scan Networks support the system's dependability throughout its whole lifetime. The knowledge about the structural and functional properties of RSNs is essential to developing an automated integration framework. Therefore, the first two parts of this chapter provide basic information about RSN structure and operation. It is impossible to analyze and enhance such dependability aspects of RSNs, as their robustness and testability without investigating possible fault locations in RSNs and defining feasible fault models. To address this problem, the third part is devoted to faults in RSNs. Finally, an efficient and scalable model is required to represent the structural and functional properties of RSNs in the developed automated framework. The last part of this chapter introduces the graph-based model of RSNs. The presented model also considers the communication between the RSN and the instruments of the underlying system.

Chapter 3

State of the Art

3.1 RSNs and their Applications

Since the early adoption of Reconfigurable Scan Networks by the standards IEEE Std. 1149.1 [IEE13] and IEEE Std. 1687, RSNs have gained an increasing interest, which has resulted in an extensive number of academic and industrial publications. First efforts to formalize modeling and verification of RSNs, and to perform test pattern generation are presented in [BKW15b]. In [ZLJ^{+}14], an approach for RSN design automation, based on the construction of SIB trees, is provided.

Over the recent years, the application area of RSNs has expanded a lot. To eliminate the necessity of using an Automated Test Equipment (ATE) to operate RSNs, in [IK16], an on-chip retargeter is introduced. It is used to operate hierarchically-structured RSNs. Recent standard proposal P1687.1 [Por16, CVTR20, vSVTR^{+}21] extends possible access interfaces over regular JTAG Test Access Port and defines access to RSNs through alternate interfaces, such as SPI or I^2C. [GWD18] provides a broadcast-based approach for RSNs, which exploits newly presented Parallel Segment Insertion Bits (PSIB) to minimize the overall access time. An approach to access general RSNs via functional interfaces has been presented in [LMZ21]. With the adoption of self-aware systems [KW18], the application area of RSNs is extended over the offline phase of the system lifecycle. Nowadays, RSNs are also used to enhance the dependability of devices during the in-the-field stage. Some important applications of RSNs include fault handling and health monitoring [ZNL16, SDJ16, TJSD18, IK19, ZC20]. However, to efficiently support the dependability management of systems, RSNs themselves have to remain dependable [KW16]. The remainder of the chapter presents the state-of-the-art solutions to analyze and enhance specific dependability aspects of RSNs.

3.2 Robust Access via RSNs

Robust access to a major part of an RSN can be obtained, if faults in RSNs are tolerated by exploiting redundancies or avoided by reducing the probability of a defect occurrence. The rest of this section summarizes the existing solutions.

3.2.1 Fault Removal

To provide a reliable access to the instruments in a defect-free case, mismatches with the specification must be mitigated at design phase or detected during RSN post-silicon validation. In [DJST19], a post-silicon validation technique is presented, which identifies possible mismatches between the specification and the actual silicon implementation of RSNs. Further on, in [DJP$^+$19], this method has been improved to consider equivalence between the structural description of an RSN and its Register Transfer Level implementation by means of simulation. In the same publication, a method to automatically generate an RTL description of an RSN based on the Instrument Connectivity Language (ICL) description is presented, which allows to correct the mismatches in the RTL description.

During the operation time, RSNs can be affected by defects. In [DSGJ19], NBTI-induced aging of RSNs is analyzed and a technique to mitigate aging effects in RSNs based on workload rebalancing is presented.

3.2.2 Fault Tolerance

Transient faults in RSNs can be efficiently tolerated by exploiting time redundancy, such that a failing access pattern is repeated again. However, this is only possible if transient faults are detected by an online test during the operation.

RSN designs, which are constructed based on parallel SIBs [GWD18], are intrinsically robust and degrade gracefully, since losing access to one branch does not result in losing access to the other parts of RSNs. However, losing the accessibility even to a single runtime-critical instrument, such as an AVFS block may already result in a system failure. For conventional parallel SIB-architectures, it is not possible to consider the criticality of such instruments. Moreover, such RSN designs are not suitable for a wide range of systems, which do not require accessing a large portion of instruments in parallel.

For arbitrary RSNs, redundancies can be exploited to provide robust access in the presence of defects. Using conventional approaches to tolerate permanent faults for the entire RSN, such as Triple Modular Redundancy (TMR) [LV62], requires high hardware costs. In [BKW20], the costs are reduced, and single faults in RSNs are tolerated by augmenting the initial RSN with additional connectivities. If a fault is activated in an RSN, it can be detected and the exact fault location can be extracted by the means of diagnosis. Faulty parts of an RSN can be isolated, such that the rest of an RSN still would remain accessible [LXM21]. However, this approach requires diagnostic support [LXM21], complicates routing for test and diagnosis [BKW15b, CSSS18, DJST19, CDRS20], requires a new retargeting in the presence of a fault and does not consider the criticality of the components.

3.3 Test and Diagnosis of RSNs

Faults in conventional scan chains can be detected by using flush test sequences shifted through a scan chain, which tests the integrity of the scan cells and their interconnection. Typically used flush patterns [LB90] include all ones, all zeros as well as the "0011" ("1100") sequence repeated to cover the whole length of the scan chain under test. The test sequence, which has been shifted into the tested scan chain, is compared with the sequence at the scan-out of the chain. If the output sequence is different from the expected one, the scan chain is faulty. For conventional scan chains, flush patterns detect certain stuck-at-faults, transition-delay faults, or broken scan chains. However, already conventional scan chain testing is challenging [MM97] [YCD$^+$08] [CMR$^+$19] if the faulty effects in scan chains and control signals require more sophisticated fault models than the stuck-at-fault assumption.

Compared to conventional scan chains, testing of RSNs is even more challenging due to the high sequential depth of RSNs, distributed control structure, as well as the complex sequential and combinational dependencies [BKW15b]. Specific fault effects in RSNs are observable only for certain RSN configurations, and sequential test pattern generation (TPG) for such faults is unfeasible for the existing TPG tools. To test an RSN, a set of test sequences is generated, such that each scan primitive of interest is tested at least once. Each test sequence in the set contains the values which are shifted into an RSN. Two types of sequences exist:

- *Test access sequences* are used to load the control scan segments of the RSN with the required control values to bring the RSN into the desired scan configuration.
- *Test workload sequences* are used to test the scan primitives located on configured scan path.

The existing RSN test methods are summarized below with respect to the fault locations.

3.3.1 Test of Scan Interfaces

In [KBSW16], the primitives located at the capture- and update- paths of scan segments are tested such that read and write operations with opposite values are performed for each segment on the active scan path. However, it is not possible to test the primitives located at the interfaces independently from the values, which are stored at the connected instruments. In realistic designs, the value of the instrument, may not be controllable or observable. In this case, it is impossible to test the scan interfaces. Moreover, the existing test methods do not consider testing reset lines of shadow registers.

3.3.2 Test of Control Primitives

Numerous works in the past have presented methods to detect certain faults in control primitives and control lines. In [KBSW16], the conditions for activating faults, which may alter or break an activated scan path, are formally analyzed with a help of a deterministic test pattern generator. The generator is able to test the faults in the combinational elements on the scan path, which are located between two adjacent scan segments, but might not be scalable due to the high sequential depth of an RSN. In [CMR$^+$15], a first method is presented to test those shadow registers, which guide the operation of control primitives, such as SIBs and scan multiplexers. In [CZP$^+$18], the reconfiguration modules themselves, such as scan multiplexers, are targeted. A method is presented for smaller RSN designs to minimize the test application time while detecting faults in the control primitives. In [CSSS18], the scalability of the test method above has been significantly improved by presenting a scalable evolutionary heuristic. In [CDRS20], the test method above has been used as a basis for an efficient diagnostic procedure for permanent faults in the reconfiguration logic. An approach from [HHD20] performs access time optimization for RSNs located in multiple power domains, and [HHD21] enhances the method above in terms of scalability. In [ZLYC20], a method is presented to decrease test application time by the means of design optimization of RSNs, which are based on parallel SIBs.

The above mentioned test, validation and diagnosis methods rely on the fact that a fault or a mismatch is detected based on the altered scan path length [BKW15b, UKW17, KBSW16, CZP$^+$18, CSSS18, DJST19, DJP$^+$19, CDRS20]. However, if such fault does not alter the length of the activated scan path, it remains undetected. Although, the untestable mismatches can be enumerated using simulation-based techniques as in [DJST19], a systematic solution to detect them does not exist in the state-of-the-art, and will be presented in Chapter 5.

3.3.3 Test of Scan Segments

In [CMR$^+$15], the scan shift logic of a scan segment is tested by configuring an active scan path such that this segment is selected and by applying flush sequences. The faults are detected by shifting a flush sequence into an activated path and observing the output sequence at the scan-output. If the expected sequence is shifted out, the scan path is fault-free, otherwise it is faulty. For "stuck-at-faults", flush sequences such as “01100” or “10011” are used for the integrity test of scan cells. Such a sequence generates all possible transitions, including "00", "01", "10", and "11". The flush sequences are modifiable for more complex fault models, such as delay faults.

Conventional methods [UKW17, KBSW16, CDRS20, CSSS18, HHD20] test RSNs once, after the manufacturing, or perform online concurrent test of RSNs [WLA$^+$21]. To avoid fault accumulation, concurrent test has to be complemented by periodic test, since components not used for some time may be still subject to aging [LSW20]. The need for periodic test is especially strong in safety critical applications like automotive [KMM$^+$21, EZ17].

3.4 Security Compliance of RSNs

Security properties of RSN, such as data confidentiality and integrity against unauthorized system alternations, have been investigated by numerous research publications, as discussed below in Section 3.4.1. Possible security threats, which may be introduced by improper RSN integration into the underlying systems, can be resolved, as presented in Section 3.4.2. At the same time, the existing solutions do not present a solution to integrate an RSN in a security-preserving way, such that an RSN generally does not introduce any unwanted extra-connectivity into the underlying system.

3.4.1 Security Analysis and Specification

Attacks using DfT infrastructure, such as conventional JTAG scan chains are well-investigated in the literature [RDN$^+$14]. A few real examples include but are not limited to the attacks on XBOX 360 [XBO], and on iPhones [Gre]. The RSN integration can introduce additional connectivities into the design, which might be exploited by an attacker as a side-channel to leak or manipulate sensitive data or alternate the system behavior [EKC21, KDD16, RBT18, KBW17]. The secure integration of RSNs is even more challenging compared to conventional scan chains [YWK06, LTPP07] due to the complex control dependencies [BKW15b]. Fully denying access to RSNs during the functional mode is not an option, since often the dependability instruments must be available online [TJSD18, IK19].

Security threats may exist due to unwanted access through the Test Access Port (TAP), but also due to the connectivities through RSNs, which allow to propagate the data between trusted and untrusted parts of an system or the pins of the chip [KBW17].

Security properties of RSNs can be validated via simulation to check the data integrity by adding hash functions to shift sequences as in [KD18], and to provide an evidence about unauthorized access attempts. Machine learning-based techniques can be used to detect attacks on RSNs [RBT18].

Security properties of RSNs, such as data confidentiality, can be verified by Craig interpolation [KBS$^+$16]. A method from [RPB20] serves as a guidance for a DfT integrator and identifies the security weaknesses of RSNs. It is possible to check, if a certain prohibited connectivity is sensitizable in the RSN [RKA+18] or through a path [RTB+19], traversing both the system and the RSN. However, a possibly exponential number of paths must be analyzed sequentially to generally verify the compliance of the RSN with the security properties of the underlying system. This makes the existing analysis methods unscalable for large RSNs and realistic security specifications [KSRG+17]. A high sequential depth of RSNs and complex control dependencies make the existing ways to compute the functional reachability [HP13, NTKW98, NSV+17] unfeasible.

3.4.2 Security-Oriented Integration of Scan Chains

Multiple solutions exist to mitigate unwanted connectivities through a test access infrastructure, such as conventional scan chains or RSNs. In conventional scan chains, a separate "secure" mode can be established to perform the confidentiality-critical computations [YWK06]. The shifted data can be encrypted as in [DFDR19] or obfuscated as in [LTPP07]. An unauthorized transition from the functional mode to the test mode can be prevented as in [DDFR13] to mitigate the test mode attacks. For a comprehensive review on scan chain security refer to [LLY+19].

Encryption and obfuscation can secure the RSN access throughout the life-cycle [LA15, TFR+19]. Fine-grained schemes, such as Locking Segment Insertion Bits (LSIBs) [ZDCP14], Parallel LSIBs [GCDE17], and Secure SIBs (SSIBs) [BKW15a], extremely complicate an unauthorized access to specific RSN parts. [PRML20] presents a fine-grained dynamic technique to protect RSNs. In [EKC21], shadow registers and information-flow tracking logic is added to prevent data sniffing and alteration. In [KBW17, DT19], additional hardware is used to build extra paths to prevent sniffing and spoofing through RSNs.

However, all the schemes mentioned above do not guarantee to prevent possible information leakage due to the RSN integration. Additional connectivity, which exceeds the allowed connectivity of the design, should not be introduced by RSNs. Such connectivities must be cut either functionally by using sequence filters, or structurally by resynthesizing some parts of the RSN. Filters [BKW14, AKS+18] enable flexible protection for multiple users with varying security requirements. However, in some cases, filters have unwanted side effects and block the access of uncritical segments as well. Resynthesis resolves all the violations structurally, and allows to preserve the accessibility to other instruments, but implies hardware overhead due to the structural changes. Moreover, it is not possible to consider multiple users with different access rights. The existing approaches [RKA+18, RTB+19] resolve the violating connectivities locally, by considering every single violation independently. This incurs many structural changes, especially if the number of violations is high as in [LAR+19] .

3.5 Summary

Analyzing the dependability properties of RSNs and, in particular, enhancement of these is far more challenging compared to the conventional scan chains due to the following properties of RSNs:

- *Reconfigurable Scan Networks are inherently sequential with a high sequential depth:* A single operation applied to an RSN might require thousands of shift cycles. At the same time, to access a particular target register, multiple reconfigurations between operations might be needed. Each reconfiguration requires analyzing not only combinational control logic but also sequential dependencies of the scan cells.

- *The existing algorithms for RSNs face scalability issues and often do not consider arbitrary RSNs:* Using conventional cycle-accurate algorithms and flat graph representations of RSN is not scalable with the realistic size and complexity of RSNs. To overcome the scalability issues, many of the existing publications concentrate on some specific types of RSN or have certain assumptions about the RSN structure [ZLJ+14, BKW15b, UKW17, KBSW16, CZP+18, CSSS18, DJST19, CDRS20], or consider the dependability properties locally, for certain pairs of vertices [RKA+18, RTB+19]. To prove some properties of RSNs in general, even using transaction-level models, such as [BKW15b] is not versatile enough. Therefore, to analyze and enhance the dependability of RSNs more sophisticated algorithms and efficient models of RSNs are needed.

- *Dependability properties of RSNs are contradicting:* Some dependability properties, such as testability of an RSN and its security compliance with the underlying device are contradicting [RDN+14]. High observability and controllability of scan primitives are required to ensure fault detection in RSNs. At the same time, extra-connectivities introduced via RSNs can be used as a side channel to leak some sensitive data. Therefore, to synthesize dependable RSNs, dependability properties have to be considered together and their interdependencies must be investigated as well.

Extensive academic and industrial research has been conducted in the field of dependable RSNs. At the same time, a comprehensive integration approach for RSNs does not exist, and the major challenges above remain unresolved for specific dependability properties. The facts above motivate the importance of an integration methodology for dependable RSNs, which is presented further in this document.

Part II

Dependability-Oriented Integration of Reconfigurable Scan Networks

Chapter 4

Robust RSNs

This chapter presents the first solution in literature to synthesize robust RSNs. Robust RSNs do not only enable an efficient *post-silicon validation* with reliable access to the significant part of the instruments but also provide *runtime* access to the most critical instruments.

The method below has been first published in [LWW22c]. First, an exact analysis is presented, which assesses the criticality of a fault in any RSN primitive for the system operation. Based on the criticality analysis, a minimized number of RSN primitives is selected and hardened to reduce the damage caused by defects. An evolutionary algorithm [ZLT01, DPAM02, Pim17, Tei12] investigates a trade-off between reducing the hardware costs of hardening and minimizing the remaining damage of defects and generates close-to Pareto-optimal solutions.

The remainder of this chapter is organized as follows. First, the goals of a robustness-enhancing RSN resynthesis are summarized in Section 4.1 and the most important criteria for integration of robust RSNs are defined. Then, in Section 4.2, the problem of criticality analysis of RSN primitives is formulated and a scalable method for so-called series-parallel RSNs is presented. A selective hardening scheme for series-parallel RSNs is developed based on the analysis results and is presented in Section 4.3. Section 4.4 shows how an arbitrary RSN can be modeled as a series-parallel one. Thereby, all the developed methods for series-parallel RSNs are applicable for arbitrary RSNs. Finally, Section 4.5 provides an overview of the evaluation results, which show the efficiency and the scalability of the developed robustness enhancement method.

4.1 Goals of Robustness-Enhancing Resynthesis

The resynthesis scheme presented below has the following goals:

- *Precise Criticality Analysis*: The criticality of scan primitives should be carefully assessed, and the most critical primitives in the RSN for the correct system operation should be identified, as shown in Section 4.2.
- *Cost-effective Selective Hardening*: The scheme should dramatically decrease the damage for the system operation due to the defects in RSNs, while minimizing the hardware costs, as shown in Section 4.3. The trade-off between the hardware overhead and the remaining damage must be investigated.
- *Access Patterns Compatibility*: The resulting RSNs should follow the topology of the initial RSNs. Thereby, they should not only be compatible with the existing access, test and diagnosis procedures [BKW15b, UKW17, CSSS18, DJST19, CDRS20], but also be able to use the same access patterns as the initial unhardened RSN.

Example: *In Fig. 4.1, an example RSN from Fig. 2.1 is presented. Assume that the accessibility of the instrument* i_3 *is critical for system operation. Imagine that the multiplexer* m_1 *is a regular scan multiplexer, and* m_2 *and* SIB *are hardened.*

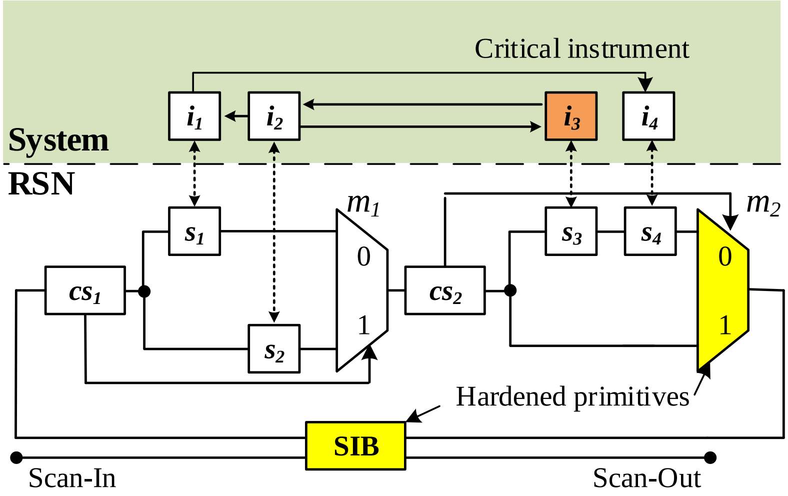


Figure 4.1: Hardened RSN Example

From the top-level-view, the RSN remains the same. In this case, the scan multiplexer m_1 *might propagate the data from its wrong input, while the multiplexer* m_2 *and the* SIB *remain* *functionally correct. Faults due to a defect in* m_2 *are avoided and the data from the critical instrument* i_3 *remains accessible.*

4.1.1 Optimization Criteria

By developing an automated framework for robust RSNs, it is first required to define the most important optimization criteria. The main aim of using Reconfigurable Scan Networks is to provide a flexible and efficient access to the embedded instrument reliably throughout the whole lifetime. Therefore, most of the instruments should remain accessible, even in the presence of defects. At the same time, the hardware overhead due to RSN integration must be minimized. Since the optimization criteria above are contradicting, multi-objective optimization methods can be used.

Multi-Objective Optimization

A *boolean optimization problem* can be formulated as follows [BV]:

$$\begin{aligned} &\underset{x \in X}{\operatorname{argmin}}[f(x)] \\ &g_i(x) \leq 0, i = 1, \dots m \\ &h_j(x) = 0, j = 1, \dots p \end{aligned} \tag{4.1}$$

where x is an n-variable vector of binary variables; $f(x)$ is the objective function to minimize in range X; $g_i(x) \leq 0$ are inequality constraints and $h_j(x) = 0$ are equality constraints.

In an optimization problem instance, multiple objective functions might have to be considered simultaneously. In this case, the problem instance is referred to as a multi-objective optimization (MOO) problem instance. A naive way to address multiple criteria in one problem instance is to merge several initial objective functions into a single objective function with a help of weight coefficients:

$$f(x) := \sum_{i=1}^{N} c_i * f_i(x) \tag{4.2}$$

where $f_i(x)$ is one of the inital objective functions; c_i is a weight coefficient showing the importance of the corresponding objective function for the resulting solution; f_x is the resulting objective function to optimize.
However, applying such a strategy would not help exploring the whole solution space, especially if the initial optimization criteria are contradicting [Pim17]. Instead, a diversified set of solutions can be generated, and the relations below are considered:

- If some optimization criteria in a solution sol_a has a better value compared to some other solution sol_b, while all the other criteria of sol_a are better or equal to the ones of the solution sol_b, then the solution sol_a dominates the solution sol_b.
- A set of so-called pareto-optimal solutions contains only dominant solutions.

The Pareto-optimal solutions to the problem form a Pareto front, as shown in Fig. 4.2, where the solutions shown in green dominate the solutions in yellow.

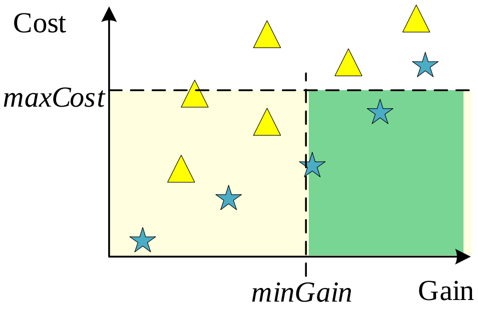


Figure 4.2: Pareto front

Evolutionary algorithms, and in particular genetic algorithms [KCK21], can be used to generate pareto-optimal or close-to-pareto-optimal solutions. Few examples of genetic algorithms include but are not limited to Non-dominated Sorting Genetic Algorithm (NSGA)-II [DPAM02] and Strength Pareto Evolutionary Algorithm (SPEA)-II [ZLT01].

Robustness of an Access

An RSN should provide a reliable access to the instruments even in the presence of defects. Loosing the accessibility to certain instruments may have a huge impact on the system operation and even lead to a system failure. Therefore, the damage caused by the inaccessibility of a certain instruments due to defects in an RSN can be reflected by an explicit criticality specification. Each instrument i is associated with a pair of non-negative *damage weights*. The first weight $damage\_obs(i)$ defines the *damage* of losing the *observability*, while the second weight $damage\_set(i)$ represents the *damage* of losing the *settability* of the instrument i. The $damage$ of a primitive p is calculated as a weighted sum of the instruments, which become inaccessible due to the fault in this primitive:

$$damage(p) = \sum_{i \in I} damage\_obs(i) * y_{i,p} + \sum_{i \in I} damage\_set(i) * z_{i,p} \tag{4.3}$$

where $y_{i,p}$ equals to 1 if the instrument i would become unobservable, when the primitive p is defect; $z_{i,p} := 1$ if the instrument i is not settable due to a defect in the primitive p. The exact values of the *damage weights* are specified depending on the needs of a system designer, e.g.:

- *Sensors*: A relatively low positive value of $damage\_obs(i)$ can be assigned to the damage caused by unobservability of one of many interchangeably used sensors. If multiple sensors become inaccessible, the damage is more severe and is calculated as a sum of the corresponding sensors' weights. Usually, the settability of sensors is not required, and the corresponding value of $damage\_set(i)$ can be set to zero or close-to-zero.

- *Runtime-adaptive instruments*: The settability of a runtime-critical instrument, e.g. AVFS control, is important for a correct operation of a device, and the corresponding damage weight $damage\_set(i)$ is set to a relatively high value. The damage due to the unobservability of such an instrument $damage\_obs(i)$ is relatively low.

The robustness of an RSN can be assessed as an overall damage to the system operation, given a defect in one of the scan primitives, as shown in Formula 4.4:

$$\sum_{p \in P} damage(p) \tag{4.4}$$

If a *maximized number of instruments* remains observable and settable through the RSN even in the presence of defects, the total *damage* to the system operation is *minimized*. The total damage to the system operation can be minimized by means of selective hardening. If a scan primitive is implemented by using a high yield cell, the probability of a fault therein is dramatically decreased. Thereby, more instruments remain accessible through an RSN and the total damage to the system operation is minimized.

Hardware Overhead

Design-for-Test structures occupy a significant area in modern circuits, and increase the overall power consumption to a large extent. Additional costs may arise after enhancing the dependability by means of resynthesis and are refered to as relative hardware overhead. While enhancing the dependability, it is necessary to minimize the relative hardware overhead as much as possible without sacrificing the RSN functionality.

To compute the relative hardware overhead, first the hardware overhead of the initial RSN is computed. Formula 4.5 shows the overall hardware overhead of an RSN:

$$\sum_{p \in P} cost(p) \tag{4.5}$$

where $cost(p)$ denotes the hardware overhead of a scan primitive p. For each scan primitive, its cost depends on the complexity (e.g., a 2-input mux vs. a 4-input mux), type of a cell (e.g., a regular cell vs. a high-yield cell [Dom12]), or length (1 scan cell vs. 100 scan cells in a scan segment).

To reflect the changes due to the resynthesis, the relative hardware overhead is calculated with respect to the overhead of the initial RSN.

$$\sum_{p \in P} cost(p)^{new} - cost(p)^{old} \tag{4.6}$$

where $cost(p)^{new}$ denotes the hardware overhead of a scan primitive p in the newly constructed RSN, while $cost(p)^{old}$ is the overhead of the same primitive in the initial RSN.

The resynthesis procedure is efficient with respect to the overhead, if the overall relative hardware overhead of the resulting RSN is minimized. The following cases are considered:

- *Additional scan primitives:* If extra scan primitives are injected into an RSN to build new scan paths, the hardware overhead is increased. The overhead of a newly injected scan primitive p_j in the initial RSN $cost(p_j)^{old}$ is equal to zero.
- *Removed scan primitives:* If some scan primitives are removed from an RSN, the overall hardware overhead decreases. The overhead of a removed primitive in the resulting RSN $cost(p_j)^{new}$ equals to zero.
- *Replaced scan primitives:* Relative hardware overhead increases, e.g., if the length of a scan segment is increased, or if a scan primitive is built by using high-yield cells instead of using regular cells. If some primitive p in an RSN is implemented by using a different type of cells, the RSN structure would remain the same, but the hardware overhead will change ($cost(p)^{new} - cost(p)^{old}$).

The remainder of this chapter presents a robustness enhancement method, which identifies the most important primitives in RSNs and replaces them with high-yield cells. Using high-yield cells significantly decreases the probability of a fault in a scan primitive and, as a result, increases the accessibility of the instruments through the resulting RSN in the presence of defects [PK00, MZS$^+$07, Dom12]. However, the relative hardware overhead increases if more primitives are hardened. Therefore, a tradeoff between the hardware overhead and the remaining damage to the system is investigated.

4.2 Criticality Analysis

The goal of this section is to assess the impact of defects in an RSN on the accessibility of the instruments and also on the system operation. The damage of a defect in a primitive p is calculated as a weighted sum of the instruments, which become inaccessible due the fault in this primitive. The damage value can be calculated as shown in Eq. 4.3.

The *damage* of losing the *observability* $damage\_obs(i)$ and the *settability* $damage\_set(i)$ are provided in the explicit criticality specification, as shown in Section 4.1.1. Identifying the values of $y_{i,p}$ and $z_{i,p}$ for all the primitives and all the instruments is computationally-intensive due to the high sequential depth and the complex control dependencies of RSNs. For so-called series-parallel RSNs this computation is significantly simpler.

The remainder of this section formally defines series-parallel RSNs in Section 4.2.1. Section 4.2.2 discusses fault effects in scan primitives on the graph connectivity. Finally, Section 4.2.3 describes the criticality analysis method for series-parallel RSNs, which considers both explicit system criticality specifications and specific fault effects.

4.2.1 Series-Parallel RSNs

Definition 4.1 (Series-parallel graph). Let $G := (V, E)$ be a directed acyclic graph with the vertex set V, the edge set $E \subset V^2$, a single source $sc \in V$ and a single sink $si \in V$. G is called series-parallel (SP), if one of the following statements is true:

- $V = \{sc, si\}$

- G is a parallel composition of two series-parallel graphs $G_1 := (V_1, E_1)$, $G_2 := (V_2, E_2)$:

$$\begin{aligned} V &:= V_1 \cup V_2 \\ E &:= E_1 \cup E_2 \end{aligned} \tag{4.7}$$

$$\begin{aligned} sc &:= sc_1 = sc_2 \\ si &:= si_1 = si_2 \\ V_1 \cap V_2 &= \{sc, si\} \end{aligned} \tag{4.8}$$

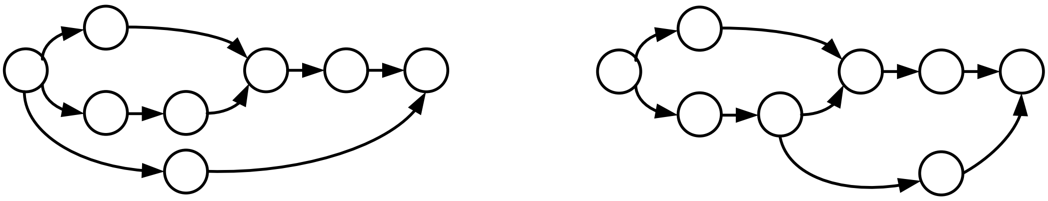
  sc_j and si_j are sources and sinks of G_j; $j = 1, 2$.

- G is a series composition of two series-parallel graphs:

$$\begin{aligned} V &:= V_1 \cup V_2 \\ E &:= E_1 \cup E_2 \end{aligned} \tag{4.9}$$

$$
\begin{aligned}
sc &:= sc_1 \\
si &:= si_2 \\
si_1 &= sc_2
\end{aligned}
\tag{4.10}
$$

Any directed graph, which does not fulfill the conditions above is referred to as a *non-series-parallel graph*. Fig. 4.3.a shows an example of a series-parallel graph, and Fig. 4.3.b shows an example without the series-parallel property.


a) Series-parallel graph

b) Non-series-parallel graph

Figure 4.3: Series-parallel property examples

The hierarchical relations are stored in a binary decomposition tree, as shown in Fig. 4.4 for the running example. The parental multiplexers are located higher in the hierarchy than the children. The tree is built bottom up, starting with the leaves. Each leaf corresponds to a vertex in an RSN graph which represents a scan primitive. If two vertices are connected in parallel in the RSN graph, their connection is represented by a "P" vertex of a binary decomposition tree (shown in green in Fig. 4.4). If the vertices are connected in series, their connection is modeled by an "S" vertex (shown in blue in Fig. 4.4). The "S" and "P" vertices are also referred to as intermediate vertices of the binary tree. The damage weights of the instruments are annotated at the corresponding segments in the binary decomposition tree.

Example: *In Fig. 4.4, the vertices* s_1 *and* s_2 *are connected in parallel by the* P/m_1 *vertex. This vertex is connected in series with the vertex* cs_1*.*

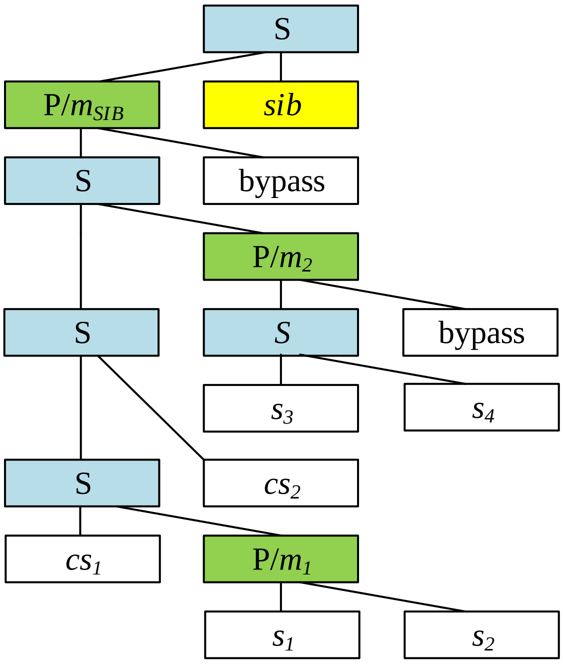


Figure 4.4: Binary decomposition tree for the graph in Fig. 2.7

4.2.2 Fault Effects in Terms of Graph Connectivity

A single fault in an RSN may affect the intended connectivity properties of the instruments and may make the RSN disconnected. This subsection discusses the influence of specific faults on the accessibility of the RSN primitives and the instruments in terms of graph connectivity. In the following, "a vertex v_j corresponding to a scan primitive p_j", will be referred to as "a primitive p_j" without loss of generality.

Scan Interface

At the high level, a fault f in a capture-circuitry prevents from reading the data from the instrument. Such a fault is modeled by removing from the binary decomposition tree the edge, which points from the scan segment to the instrument. A fault in an update-circuitry may corrupt writing to the instrument, and is modeled by removing the edge from the instrument to the scan segment.

Scan Multiplexer

If a scan multiplexer is affected by a "stuck-at-0" or a "stuck-at-1" fault f, the opposite branch of this multiplexer becomes inaccessible through the multiplexer. For all the primitives located in this branch, a path may be corrupted. To model this fault, the connectivity from a vertex, which corresponds to a faulty multiplexer, to the inaccessible branch is removed from the binary decomposition tree.

Scan Segment

A fault f affecting a scan segment may break the integrity of all the scan paths, which traverse this scan segment. The existence of a faulty scan segment is modeled by annotating the vertex, which corresponds to the affected scan segment, with the corresponding fault effect. An example here could be a flip-flop transparency fault, which decreases the length of the corresponding scan segment. In this case, the corresponding vertex is annotated with the decreased length instead of the original length.

If the scan segment becomes inaccessible due to a fault f, the fault effect can be isolated inside the branch of the RSN, which is controlled by the closest parental scan multiplexer of the given scan segment. The parental scan multiplexer is identified by traversing the binary decomposition tree of the RSN in a reversed Polish order starting from the affected scan segment. In the isolated branch, the controllability of those segments is affected, which are located closer to the scan-output than the affected segment. In a binary decomposition tree, it is equivalent to removing the connectivity to the affected vertex. This modified tree is further referred as a *settability tree* under a *fault* f. The same idea is applied to build a *observability tree* under a *fault* f. The segments, which are located closer to the scan-in, become unobservable and are disconnected in the observability tree.

Example: *Assume the SIB from the running example is affected by a "stuck-at-1" fault. This fault is modeled by removing the edge from the decomposition tree, as shown in Fig. 4.5. The instruments $i1$, $i2$ and $i3$ become inaccessible through the RSN for observation and control.*

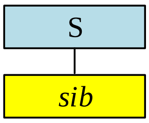


Figure 4.5: Faulty tree from Fig. 4.4, the SIB is affected by a stuck-at-1 fault

4.2.3 Criticality Analysis for Series-Parallel RSNs

The value $damage(p)$ of a primitive p (Eq. 4.3) is computed recursively, starting from the lowest left node of a binary decomposition tree and follows the order of a reverse Polish notation. Thereby, it is ensured that the relative criticality computation for those primitives which are located at the lower levels of the tree starts before the computation of their parents. For series-parallel RSNs, the values of $y_{i,p}$ and $z_{i,p}$ can be efficiently assessed:

- For a defect in a primitive p, the modified tree from Section 4.2.2 is considered. The value of $y_{i,p}$ is set to 1, if the instrument i is disconnected from the primitive p in the observability tree. Otherwise, it is set to 0. The same logic is applied to compute the value of $z_{i,p}$ with a help of a settability tree.

- If a primitive p is hardened, a fault f is avoided and the initial decomposition tree is used for assessing the values of $y_{i,p}$ and $z_{i,p}$.

As a result of the criticality analysis, a prioritized list of the primitives is generated. In this list, each primitive is associated with a possible damage. This list together with a binary decomposition tree is used for criticality assessment of hardened RSNs during the selective hardening phase, which is described in the next section.

Example: *In Fig. 4.1, the criticality of the scan multiplexer m_1 is determined as a sum $\{d_{i1}+d_{i2}\}$, since a fault affecting this mux might make the underlying instruments inaccessible for observation and control. As shown in Fig 4.6, after the criticality of the the multiplexer m_1 is assessed, the whole subtree consisting of the segments s_1 and s_2 is merged to a single node and the computation continues following the order of the reverse Polish notation.*

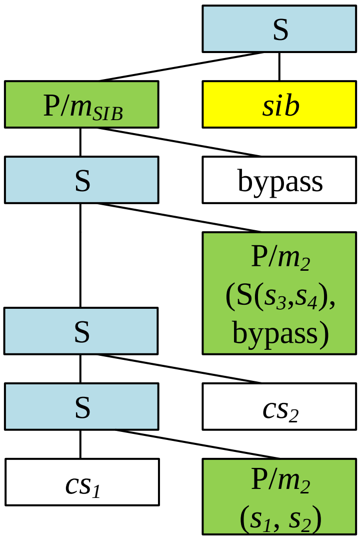


Figure 4.6: Tree from Fig. 4.4 with merged nodes

4.3 Selective Hardening for Robustness Enhancement

As soon as the most critical primitives of the RSN are identified with the help of the precise criticality analysis, and the influence of defects on the accessibility of the instruments is investigated, a selective hardening scheme is applied. This section presents the selective hardening scheme for series-parallel RSNs, and in the next section it is explained how the developed scheme can be applied for arbitrary RSN structures.

The remainder of this section is organized as follows. First, in Section 4.3.1, the selective hardening problem is formulated as a multi-objective problem instance. Next, Section 4.3.2 discusses how evolutionary algorithms are applied to solve this problem.

4.3.1 Hardening as a Multi-Objective Problem

In a selective hardening problem instance, we are looking for a set of primitives to be hardened which minimizes both the cost for hardening and the possible remaining damage due to the primitives not hardened. The accessibility of the instruments through the resulting hardened RSN in the presence of defects must be comparable to the accessibility through the original defect-free RSN. The solution should satisfy the following optimization criteria:

- A *maximized number of instruments* remains observable and settable through the RSN even in the presence of defects. As a consequence, the total *damage* to the system operation (Eq. 4.11) is *minimized*:

$$\sum_{p \in P} damage(p) \tag{4.11}$$

- The *total cost of hardening* (Eq. 4.12) is *minimized*:

$$\sum_{p \in P} \Delta cost(p) * x_p \tag{4.12}$$

  where the weight $\Delta cost(p) := cost(p)^{new} - cost(p)^{old}$ is the additional hardware overhead of a primitive p due to hardening. The variable $x_p := 1$, if the primitive p is hardened and $x_p := 0$ otherwise.

4.3.2 Evolutionary Approach for Primitives Selection

Minimizing the damage due to defects in general increases the costs of hardening. Therefore, a trade-off between these criteria is investigated by computing close to Pareto-optimal solutions. The parameter space is explored by applying the evolutionary algorithm SPEA-2 [ZLT01] of the Opt4J framework from [LGRT11]. First, the major steps of an evolutionary algorithm are described. Finally, the details are provided, how an evolutionary algorithm is applied to solve the selective hardening problem.

Evolutionary Algorithm Steps

An evolutionary algorithm is applied iteratively, and each iteration is referred to as a generation. As more generations pass, the generated solutions may move closer to the Pareto-front. In general, an evolutionary optimization algorithm includes the following steps:

1. *Read the initial problem:* The initial problem is encoded as a model, suitable for genetic programming.

2. *Generate the initial population:* A diversified set of genes is generated. Each time when a new population is generated, the number of generations is incremented.

3. *Calculate the fitness function:* The candidates are assessed by using the fitness functions. A fitness function is an optimization criteria, e.g., provided in Eq. 4.11 and Eq. 4.12. The candidates, which dominate other candidates, are kept for mating, while the dominated candidates are dropped from the list.

4. *Check the termination criteria:* If the allowed number of generations is exceeded, the computation terminates with a set of close-to pareto-dominant solutions.

5. *Generate the next population:* A limited number of individuals is selected from the current population for mating.

6. *Perform mating:* Crossover and mutation are applied to the selected individuals with a determined probability:

   - *Individual bit mutation:* A random bit is flipped.
   - *One-point crossover:* Two offsprings of length r are generated from two parental individuals. For the first offspring, n bits are taken from the first parental gene and another $r-n$ bits are taken from the second gene. For the second offspring, vice-versa.

   The computation continues from the step 3 until the fitness function stops improving better than for a threshold ε after the application of the next n generations.

Primitives Selection

The goal of the selective hardening scheme is to find a set of close-to pareto-optimal solutions with respect to the criteria formulated in Section 4.3.1. Each problem solution is an assignment of the values of $\{x_1, \ldots x_p \ldots, x_{|P|}\}$, such that x_p equals to 1 if a given scan primitive p is hardened and to 0 otherwise.

A binary decomposition tree of a series-parallel RSN graph is used for the computation. In a evolutionary algorithm formulation, each problem solution corresponds to a gene, which is implemented as a list of binary values $\{x_1 := a_1, \ldots x_p := a_p \ldots, x_{|P|} := a_{|P|}\}$.

- First, a set of diversified initial solutions to the hardening problem is generated, where a random subset of instances is hardened.
- The obtained solutions are assessed with respect to optimization criteria in Section 4.3.1. Although, we can directly control only the values of x_p, the interdependence between the values of variables x_p and $y_{i,p}$ allows to implicitly control the values of $y_{i,p}$. If a certain primitive is hardened, it implies that a defect in this primitive cannot occur and the observability of its children is not affected. The same applies to the settability of the instruments.

  For series-parallel RSN graphs, the values of the cost functions can be assessed very fast by traversing the corresponding parts of the binary decomposition tree.
- The crossover and mutation operations are applied to the dominant solutions.
- The steps 2 and 3 are repeated until the required number of generations is passed or the obtained solutions stop improving for a certain number of generations.

As a result, a diversified set of close-to Pareto-optimal solutions is generated, such that a tradeoff between the cost of hardening and the remaining damage due to the defects is investigated. The primitives, which are selected by the algorithm, are substituted by high-yield cells. In the resulting RSNs, the top-level RSN structure is not affected by the presented method. The resulting hardened RSNs are not only compatible with all the existing access, test and diagnosis procedures [BKW15b, UKW17, CSSS18, DJST19, CDRS20], but can also use the same access patterns as the initial RSNs.

4.4 Application for Arbitrary RSN Structures

The method presented above is only applicable, if an RSN can be represented by a series-parallel graph. The method presented below in Section 4.4.1 allows to identify whether the initial graph is series-parallel. Although most RSNs graphs are series-parallel, for some RSNs, additional steps might be required to obtain a functionally equivalent series-parallel graph, as shown in Section 4.4.2. After an equivalent series-parallel representation of a non-series-parallel RSN graph is constructed, this generated representation is used for performing the criticality analysis and the selective hardening scheme presented above.

4.4.1 Validation of the Series-Parallel Property

To check whether a specific RSN graph is series-parallel, first few simple checks are applied and then a reduction algorithm based on [VTL79] is used:

- *Initial Checks*

  First the reachability of all scan primitives is computed. If the initial RSN graph has multiple sinks or sources, auxiliary vertices are added into the RSN graph and serve as a pseudo-primary sink and source correspondingly.

  The acyclicity of the initial graph is validated, since a graph, which contains cycles, is not series-parallel by definition. If the initial RSN graph contains cycles, an acyclic representation is constructed by removing a few edges in a similar way as it is well-known in partial scan design [KW90].

- *Main Flow*

  The main flow of the check follows the well-known reduction algorithm from [VTL79]. If two vertices v_1 and v_2 of the RSN graph are connected in series or in parallel, they are merged into a single vertex. The vertices are merged until it is not possible to merge any pair of vertices.

  For a series-parallel graph, after the algorithm above is applied, the whole RSN graph is represented with a graph, which consists of a single vertex. If such a representation is not possible, the graph is non-series-parallel and has to be further processed as described below. The Church-Rosser property of the applied reduction system [CR36] allows to apply reductions in an arbitrary order to validate the series-parallel property.

4.4.2 Transformation into a Series-Parallel Graph

To build a series-parallel equivalent representation of a non-series-parallel graph, a minimized number of additional virtual vertices is added to the initial RSN graph [KG14]. Since the virtual changes are reverted in the resynthesis phase, additional hardware overhead is not needed to obtain a series-parallel representation of the RSN graph.

The fan-out stems are identified, which prevent the RSN graph from being series-parallel. Any fan-out stem f_{viol}, which is located in the stem region of another fan-out stem f_{init}, and which has either the same closing reconvergence gate or its closing reconvergence is reachable from the closing reconvergence of the stem f_{init}, is referred to as a violation spot. To resolve the violation, the vertices, which are located between the fan-out stem f_{init} and the violation spot f_{viol}, are duplicated and are placed after the violation stem in the graph representation. The violation spots are resolved sequentially, until a series-parallel representation of the RSN graph is obtained. The violation spots and their relative order of processing are selected in a topological order of the RSN graph. This order starts at the scan-in port. The fan-out stems, which are located closer to a primary scan-in vertex, are processed first, followed by the fan-out stems in their stem region, each time either going deeper in the hierarchy, or moving forward to a succeeding fan-out stem.

Example: *In Fig. 4.7, a connection from the vertex* f_3 *to the vertices* m_2 *and* m_3 *makes the RSN graph non-series-parallel. If it would be simplified as much as possible, a single vertex representation will not be achieved.*

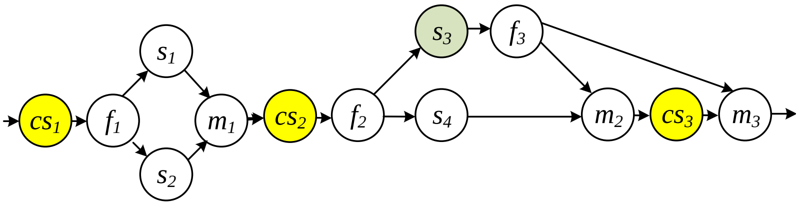


Figure 4.7: Non series-parallel graph

The decomposition tree for the resulting structure is shown in Fig. 4.8. In Fig. 4.9, an NSP region is transformed into a series-parallel form by duplicating the vertex s_3 *and the fan-out stem* f_3.

The resulting binary decomposition tree, as shown in Fig. 4.10, only contains parallel and series compositions, as well as the leaf nodes, which correspond to the individual scan segments. It can be processed to enhance the robustness, as discussed above.

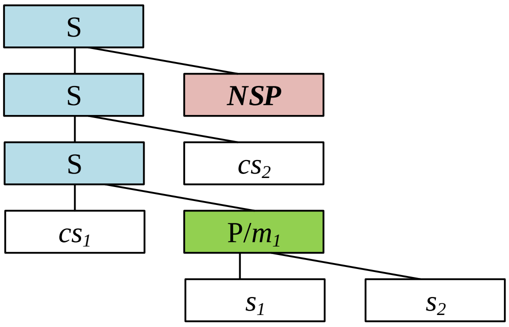


Figure 4.8: Binary decomposition tree for the non-series-parallel RSN graph in Fig. 4.7

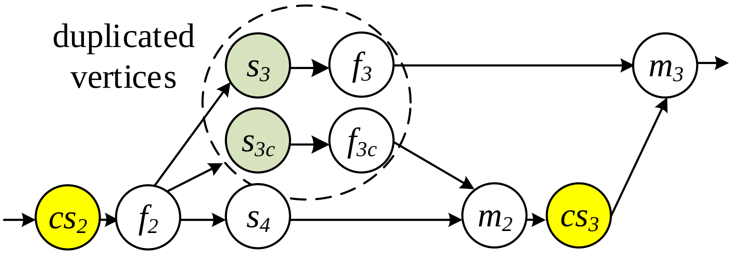


Figure 4.9: Transformed subgraph of the non-series-parallel graph from Fig. 4.7

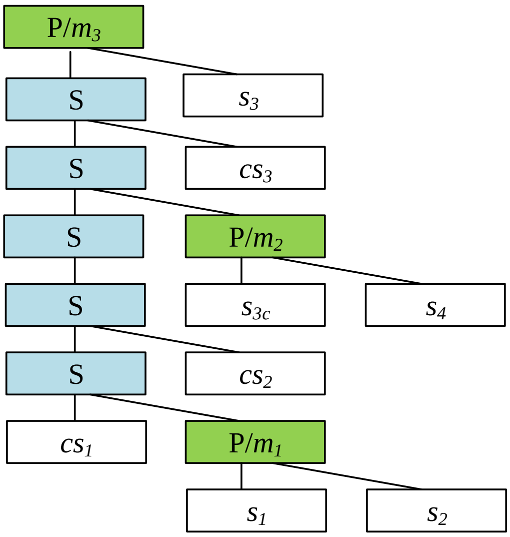


Figure 4.10: Binary decomposition tree for a series-parallel representation of the non-series-parallel graph from Fig. 4.7

4.5 Evaluation

This section presents the evaluation results for the developed robustness enhancement scheme for RSNs. The evaluation is based on the detailed data reported in Appendix B.1. The information about the benchmark sets is given in Appendix A.

To ensure scalable processing, the developed analysis and enhancement scheme is applied to series-parallel representations of RSNs. In Section 4.5.1, the series-parallel property of the RSN benchmarks is validated and, if needed, a few virtual vertices are added into an RSN graph to obtain its series-parallel representation. Section 4.5.2 presents the evaluation results for the presented analysis and resynthesis methods. The detailed experimental results for the developed methods are provided in Appendix B.1.

4.5.1 Series-Parallel Property

The series-parallel property has been checked for all the benchmarks from the commonly recognized ITC'2016 [TJD$^+$16] and DATE'2019 [RTB$^+$19] benchmark sets and a series-parallel representation has been constructed, if needed. The experiments show the following results:

- **ITC'2016:** As expected, most of the RSN benchmarks can be modeled as series-parallel graphs, and a transformation into a series-parallel form is not required. Only the "*TreeFlat*" benchmark graph is not series-parallel. It is transformed into a series-parallel form, and virtual vertices are added to perform the transformation.

- **DATE'2019:** A hierarchical structure of benchmarks from the DATE'2019 benchmark set is shown in Fig. 4.11. A top-level chip TAP controller is used to access N cores, such that each of the cores accesses the memory via M controllers. Each core is accessed via a separate Segment Insertion Bit (SIB core in Fig. 4.11). The corresponding RSN graphs are close-to series-parallel. Each sub-RSN, which corresponds to a single core, is modeled as a series-parallel graph. A top-level representation is transformed from an arbitrary non series-parallel graph into a series-parallel graph by adding just two virtual vertices for each benchmark.

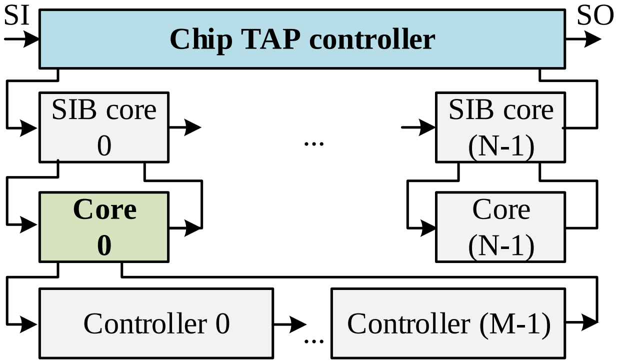


Figure 4.11: DATE'19 MBIST benchmarks' structure

The resulting series-parallel representations of RSNs support the scalable criticality analysis and resynthesis. For series-parallel RSN graphs, binary decomposition trees have been generated, as shown in Section 4.2.1.

4.5.2 Robustness Enhancement

Initial assessment

For all the benchmarks, a criticality specification has been provided. In the specification, the damage weights of the instruments are given. In the experiments, positive damage weights of loosing observability have been assigned to 70% of the instruments. Additionally, positive damage weights of loosing controlability correspond to another 70% of randomly selected instruments. Finally, 10% random instruments have been selected as important for observation, another 10% - as important for control.

For all the becnhmarks, all the instruments are accessible via an RSN, if the RSN is defect-free. The initial worst-case assessment of the damage to the system operation is performed as shown in Eq. 4.4, if none of the primitives is hardened. Each time, a defect in one scan primitive is considered. This value is used to assess the effectiveness of the developed resynthesis scheme with respect to the remaining damage. Additionally, the maximum hardware cost is calculated for the case, if all the scan primitives are substituted by high-yield cells. This value is further used as a reference to assess the hardware overhead due to hardening. In this section, the damage to the system operation is normalized with respect to the initial worst-case assessment above. Similarly, all the numbers for the hardware cost are normalized with respect to maximum hardware cost. The exact absolute values for all the benchmarks are provided in Appendix B.1.

Criticality Analysis and Robustness Enhancement

The criticality analysis from Section 4.2 has been conducted given an explicit specification. The criticality of RSN primitives has been assessed with respect to the damage to the system operation. The primitives to harden are selected by using the evolutionary algorithm called SPEA-2 [ZLT01] implemented in the Opt4J framework from [LGRT11]. Genetic algorithms produce a series of incrementally improving solutions for a given problem. For each generation, a tradeoff between the cost of hardening and the remaining damage to the system is explored. The genetic algorithm stops when each further N generations improve the generated solutions less than by a given threshold. The exact parameters of the evolutionary algorithm are given in Appendix B.1.

To demonstrate the effectiveness of the developed methods with respect to the hardware cost and the remaining damage to the system operation, two dominant solutions are shown below for each benchmark:

- Fig. 4.12 presents the best damage-reducing solution, which requires at most 10% hardened primitives. For all the benchmarks, by hardening a minor number of primitives, it is possible to significantly increase the probability that the instruments would remain accessible through the resulting RSN even in the presence of defects. The horizontal dashed line in Fig. 4.12 shows the initial assessment of the parameters.
- In Fig. 4.13, the most cost-efficient solution is provided for reducing the damage down to 10% of the initially assessed value. For all the benchmarks, the damage to the system is reduced by an order of magnitude by hardening a negligible fraction of primitives.

In both cases above, all the important instruments remain accessible via the resulting RSNs. The developed analysis and resynthesis methods are scalable with the increasing size and complexity of RSNs. In the worst case, 1.5 hours have been required to perform the initial analysis and to select the primitives to harden. For most benchmarks, the overall processing time was in the range of several minutes or even seconds.

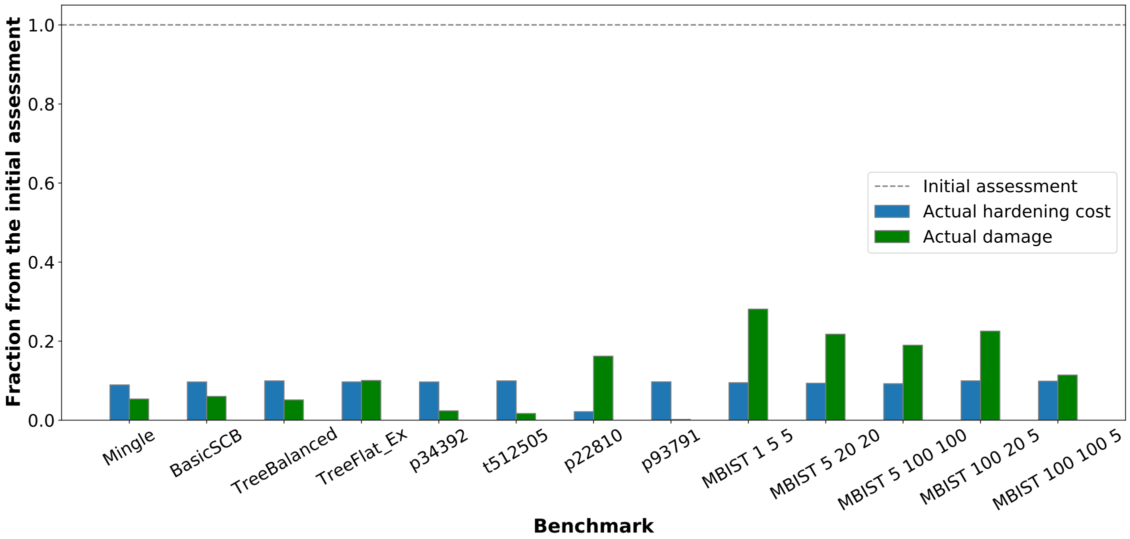


Figure 4.12: Robustness enhancement results, cost $\leq$ 10%

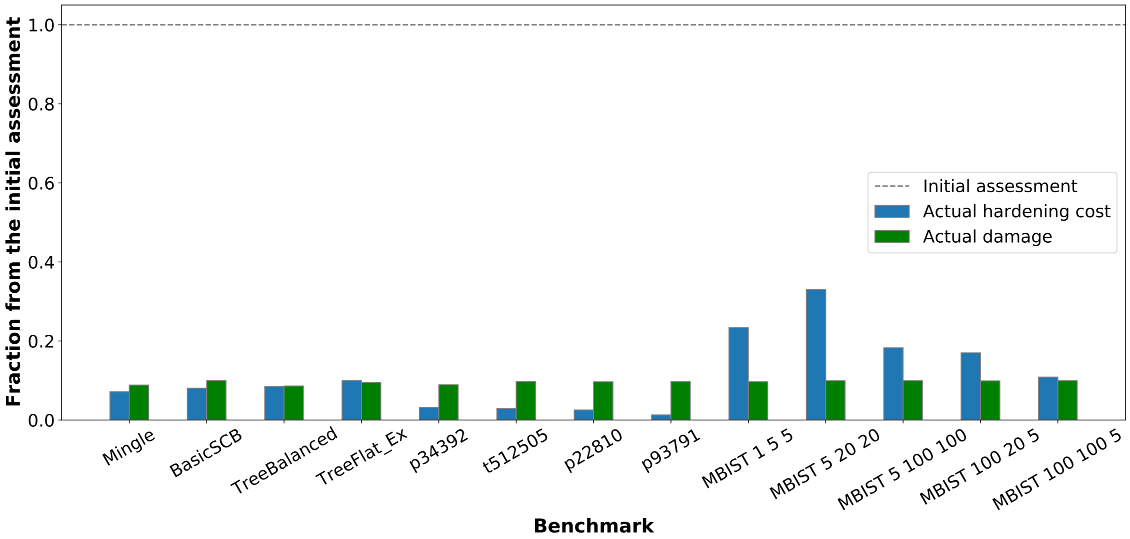


Figure 4.13: Robustness enhancement results, damage $\leq$ 10%

Case-study

Fig. 4.14 shows the obtained hardening results for one specific benchmark in more details. Here, a pareto plot is provided for the p93792 benchmark where the instances from five selected generations are presented. Initially (1st generation), a randomized subset of scan primitives is selected for hardening. As more generations pass, the solutions improve, and only dominant solutions are selected at each further generation. At 1501st generation, the solutions saturate: each further 100 generations improve the solution by at most 1%.

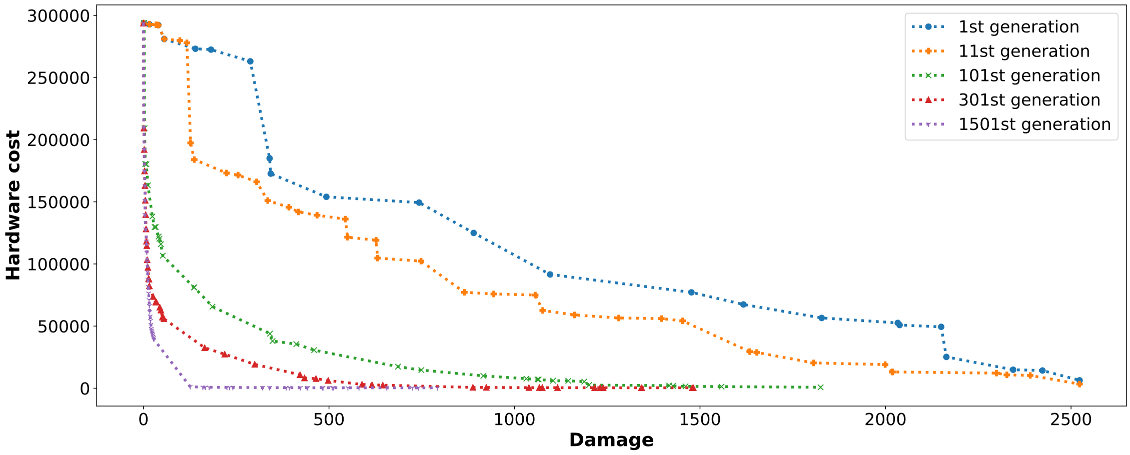


Figure 4.14: Tradeoff between the hardware cost and the damage, case-study

4.6 Summary

A single fault in an RSN may dramatically reduce the accessibility of the instruments. Hardening the most critical instruments and the corresponding scan segments against permanent faults is not enough to ensure robust access to the instruments since a single fault in the control logic of an RSN may corrupt scan paths and make certain instruments inaccessible. This will affect two major tasks:

- During *post-silicon validation*, an innovative process or a new design may show an increased defect rate. A single fault in an RSN may prevent accessing a major part of instruments, such that a large portion of the validation data cannot be extracted from a limited number of prototypes.
- During *online operation*, the device operation may be guided by runtime-adaptive instruments, examples are Adaptive Voltage and Frequency Scaling (AVFS) [TKD$^+$07], temperature control or error rate adoption. The inaccessibility of runtime-critical instruments via a defect RSN may eventually result in a system failure or even cause permanent system damage.

In this chapter, a method to generate robust RSNs is presented. The resulting RSNs ensure reliable access to the most relevant instruments throughout the device lifetime. A minimized number of scan primitives uses hardened cells of high yield. The scan primitives to harden are selected based on the precise criticality analysis. To ensure the scalable processing of RSNs, series-parallel RSN model is introduced and its applicability for arbitrary RSNs is discussed. All the critical instruments and most of the remaining instruments are accessible through the resulting RSNs even in the presence of defects. A trade-off between the hardening cost and the remaining damage of defects for the observability and the settability of the instruments is investigated by using an evolutionary algorithm. As a result, a set of close to pareto-optimal solutions is computed. The resulting RSN follows the topology of the initial RSNs, the access latency of the resynthesized RSN does not change. Thereby, the resulting RSN is not only compatible with the existing access, test and diagnosis procedures [BKW15b, UKW17, CSSS18, DJST19, CDRS20], but is also accessible by the same access patterns as the initial RSN.

Chapter 5

Testable RSNs

Reconfigurable Scan Networks can occupy a significant part of the chip area, the probability of a fault within an RSN cannot be neglected, even if the most critical parts of RSNs are hardened as discussed in the previous chapter. At the same time, due to high sequential depth and complex control dependencies, major parts of RSNs have limited observability and controlability. As a result, the existing test and diagnosis schemes are not sufficient to detect all the faults in RSNs, as discussed in Section 3.5.

This chapter presents efficient methods to significantly enhance the testability of RSNs by means of resynthesis and to test RSNs throughout their whole lifetime. The remainder of the chapter is organized as follows:

- Section 5.1 presents a complete design-for-test (DfT) scheme for RSNs, which has been first published in [LWW22a]. Design-for-Test methods ensure fault detection for three major parts of an RSN: scan interfaces, scan segments and control primitives. An efficient test integration scheme is developed, which dramatically decreases the overall test access time compared to the case, when the individual parts of an RSN are tested independently.
- In Section 5.2, an online periodic test of RSNs from [LWW22b] is presented. The developed method complements the existing test and DfT methods for RSNs and allows to avoid fault accumulation in those primitives, which are rarely accessed. A small set of test access sequences is generated for a given RSN and can be applied periodically within safety margins. The generated test sequence set can be reused during the whole lifecycle of RSNs, including also the offline phase.

5.1 A Complete Design-for-Test Scheme

A complete Design-for-Test scheme is presented below, which combines and extends the testability enhancing schemes from [UKW17, WLA$^+$21, LWW21]. It is inherently flexible with respect to the considered fault models, and requires negligible hardware overhead. The contributions of this chapter are as follows:

- The *testability of the shadow registers and the scan interfaces* to the instruments is enhanced and the faults in the capture- and update-circuity of the scan segments become detectable.
- The *testability of control primitives* is enhanced. Existing methods test the control primitives by observing the length of an activated path [BKW15b, UKW17, KBSW16, CZP$^+$18, CDRS18, DJST19, CDRS20], and fail if an erroneously activated path has the same length as the correct one. An exact testability analysis method is presented to identify all single control faults, which do not have impact on the length of the activated path. If such a fault is identified, automated resynthesis changes the length of a minor number of scan paths to ensure fault detection.
- *The test of the scan shift logic* is enabled by integrating a compact Built-In Self-Test (BIST) structure, which is responsible for generation of a short pre-sequence to test the shift logic of the currently activated scan path.
- *Test integration* is supported to reduce the resulting test time. The generated test sequences are capable to cover multiple fault locations at a time. Each sequence contains a workload sequence, which is used to test faults affecting scan interfaces and control primitives, and a short self-generated pre-sequence to test the shift logic of the segments on the activated path.

The remainder of this chapter is organized as follows. In Section 5.1.1, the detectability of faults is formally defined as an optimization criteria for a testable RSN. In Section 5.1.2, the necessity of a complete design-for-test scheme to complement the existing test methods is motivated. In Section 5.1.3, a DfT scheme is presented for shadow registers and interfaces to instruments. Section 5.1.4 presents a testability enhancement technique for control primitives. In Section 5.1.5, the details about the scan segment test are provided. Section 5.1.6 provides details about the overall test integration procedure.

5.1.1 Optimization Criteria

The main aim of using Reconfigurable Scan Networks is to provide a flexible and efficient access to the embedded instruments reliably throughout the whole lifetime. However, even if the test method is thorougly developed, certain faults may remain undetectable due to the low observability and controllability of certain fault locations. Therefore, fault detectability is an important optimization criteria for the integration of dependable RSNs. At the same time, to ensure fast and efficient access to the instruments, and also to perform tests within a given timing margin, it is important to keep the access latency low. The remainder of this section provides more details for the optimization criteria above.

Detectability of Faults

To detect a fault in an RSN it is necessary to propagate its fault effect to the scan-out port of the RSN, and distinguish the faulty output from an expected fault-free output. It can be done e.g. by observing the altered path length or altered signature [BKW15b, UKW17, CSSS18, DJST19, CDRS20]. In Formula 5.1, the detectability of faults is calculated:

$$\frac{\sum_{f \in F} testable(f)}{|F|} \tag{5.1}$$

where F is the total set of faults; $testable(f)$ equals to one if a fault f is testable by the existing test methods for RSNs; it equals to zero otherwise.

To increase the detectability of faults in a given RSN, first, undetectable faults must be systematically identified. Their testability can be increased by using a resynthesis method, as shown in further in this chapter.

Access Latency

Accessing an instrument via an RSN requires at least one CSU-operation to configure an Active Scan Path, which would include a scan segment accessing the given instrument. Formula 5.2 calculates the maximum access latency over all the scan segments:

$$\max_{s \in S}[AT(s)] \tag{5.2}$$

where $AT(s)$ denotes the access latency of the segment s in terms of clock cycles.

In a dependable RSN, all the instruments must be accessible for the eligible users, and the access latency can be minimized to provide efficient access. The maximum access latency over the segments can be minimized by resynthesizing an RSN. For example, long scan chains can be split into muliple shorter chains. To compare the access latency of a resynthesized RSN with the latency of the intial RSN, a change of the access latency over all the segments is computed, as shown in Formula 5.3:

$$sum\_latency\_difference := \sum_{s \in S}[AT^{new}(s) - AT(s)], \tag{5.3}$$

where $AT^{new}(s)$ is the access latency of the segment s in the resynthesized RSN.

Alternatively, the latency of the segment s_{max} with the highest latency in the original RSN, can be compared with the latency of the same segment in the resynthesized RSN, as shown below:

$$\begin{aligned} s_{max} &:= \operatorname*{argmax}_{s \in S}[AT(s)] \\ latency\_difference &:= AT^{new}(s_{max}) - AT(s_{max}) \end{aligned} \tag{5.4}$$

Example: *The maximum access latency among the registers of an online periodic BIST can be minimized in order to ensure an efficient periodic access through an RSN. Using this criteria results in a parallelized RSN, with a higher number of configurable ASPs.*

5.1.2 DfT as Essential Companion for RSN Test

Specifics of some RSN structures may affect the detectability of faults. In Fig. 5.1, some examples of the testability issues are presented, which would arise for the RSN example from Fig. 2.1, if the existing test methods would be applied.

1. *Undetected fault affecting a shadow register of s_3 and a faulty reset line*: If the shadow register of the scan segment s_3 is faulty, an erroneous data might be captured into the corresponding instrument i_3. The existing methods rely on the assumption, that

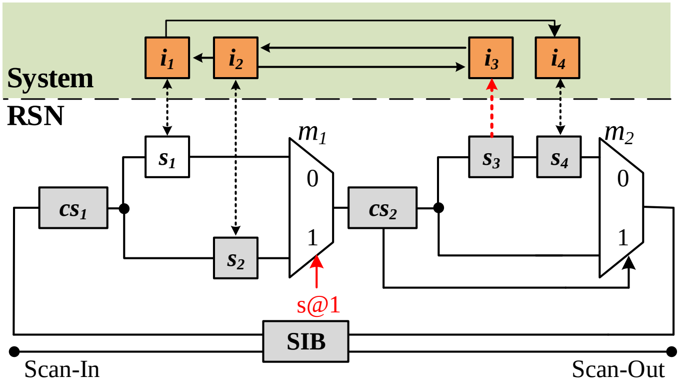


Figure 5.1: Testability issues in the RSN example from Fig. 2.1

the value in the instrument i_3 is directly observable, which is not always true. They do not guarantee to detect faults affecting the reset values of the shadow registers.

A DfT enhancement is presented in Section 5.1.3 to test the scan cell internal multiplexers ($M2$ *and* $M3$ *in Fig. 2.5) and the shadow registers at the scan interface independent of the values in the instruments. The reset line is tested as well.*

2. *Silent data corruption due to a fault at the multiplexer* m_1: If the multiplexer m_1 is affected by a "stuck-at-1" fault, a path through the grey-colored primitives would be activated in Fig. 5.1 instead of the intended path in Fig. 2.1. Since both paths have the same length, the fault at m_1 would remain undetected, and silent data corruption may arise.

   A design-for-test technique is presented in Section 5.1.4 to ensure that any single control fault results in an altered scan path length, such that the coverage of the methods presented above for faults in the control primitives is guaranteed.

3. *Corrupted scan path integrity due to the faulty segment* s_2: If the scan segment s_2 is faulty, the integrity of the configured scan path (in grey) is corrupted. Using the existing methods, it is not possible to detect this fault concurrently to the functional operation.

   A DfT enhancement is presented in Section 5.1.5 to ensure that any single fault in the scan shift logic is detectable concurrently to testing other parts of an RSN.

To overcome the above mentioned limitations of the existing schemes, the remainder of this section presents a complete design-for-test (DfT) solution for RSNs, which enhances the testability of the RSN primitives to allow complete fault coverage.
The presented scheme has the following goals:

- **Testability:** Faults affecting all parts of an RSN must be detectable, which includes instrument interfaces, scan segments and control primitives.
- **Flexibility:** The presented scheme must be adjustable towards a used-defined fault model.
- **Cost-efficiency:** The presented scheme must have a low hardware-overhead.
- **Compliance:** The DfT logic must not affect precomputed retargeting sequences.
- **Scalability and generality:** The presented scheme must be applicable to large arbitrary RSN designs.
- **Compactness:** Test sequence are supported to cover multiple test locations.
- **Compatibility with the existing test methods** The presented DfT scheme must be compatible with the test, diagnosis and post-silicon validation methods discussed above and is supposed to be used complementary to these schemes.

5.1.3 Test of the RSN Interfaces

In this section, the testability enhancement of scan interfaces between the scan segments and the instruments is discussed, which has been first published in [UKW17]. First, the problem is formulated with respect to the fault locations, which cannot be tested with the existing methods. Next, a DfT enhancement is presented to significantly increase fault coverage with respect to the faults at the scan interfaces.

Problem Formulation

The goal of the presented DfT enhancement is the test of the scan interfaces of all data scan segments. At the same time, data stored in the instruments shall not be corrupted.

As a result, the scan cell internal multiplexers ($M2$ and $M3$ in Fig. 2.5) and the shadow registers must become testable.

DfT Enhancement

The testability of the scan interfaces is improved by significantly increasing the observability of the shadow registers and decoupling the test of the multiplexers $M2$ and $M3$ from the data in the underlying instruments. With this scheme, the corresponding fault effects become observable at the scan output of a scan cell and can be propagated to the global scan-out port by using conventional test methods.

The test of scan interfaces to the instruments is enabled by augmenting the initial scan cell structure (Fig. 2.5) with an additional feedback loop between the shadow flip-flop and the scan flip-flop, as shown with a green color in Fig. 5.2.

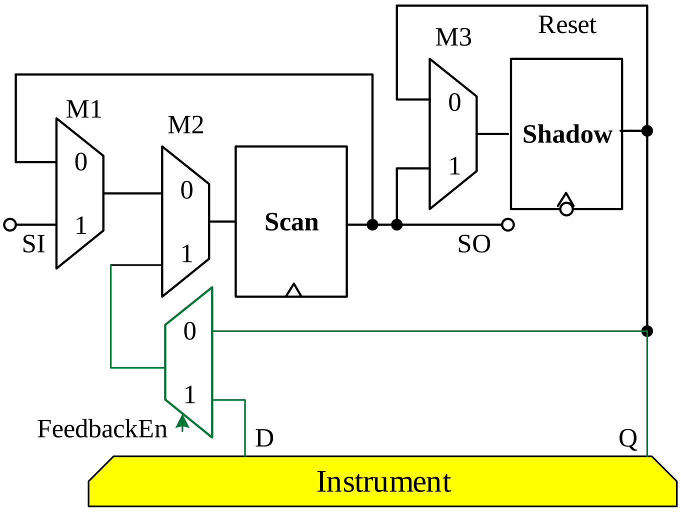


Figure 5.2: Scan cell with a DfT Enhancement (in green). The additional scan multiplexer allows to propagate the data from the shadow flip-flop to the scan flip-flop.

The DfT structure provides direct visibility of the shadow flip-flop without requiring knowledge about or control over the connected instrument. The feedback loop propagates the value stored in the shadow flip-flop into a scan flip-flop. This data is then shifted through an activated scan path, such that the value of the shadow flip-flop is observable at the scan output. The feedback loop is activated by setting the control signal *FeedbackEn* to a logic one. The scheme is compliant with IEEE Std. 1687-2014 [IEE14]. The feedback loop can be described by means of the Instrument Connectivity Language, and therefore can be readily handled by EDA tools supporting this standard. The additional feedback enable signal can be controlled externally by the access interface or internally by using previously unused assignments to the internal control signals.

In the remainder of this section, it is discussed how the DfT scheme is applied to enhance testing of the scan interfaces and the reset functionality. Since the newly integrated DfT feedback loop must be tested as well, a discussion about the testability of the corresponding faults concludes the section.

1. **Testability Enhancement for the Scan Interfaces**

   In an enhanced scan cell (Fig. 5.2), an update register can be tested by writing complementary values into the update flip-flops and reading them through a feedback loop. Faults effects residing in the update flip-flop are propagated to the scan output of the RSN with the help of the feedback path (shown in green in Fig. 5.2) and the initial paths through a scan cell by applying the following steps:

   (a) First, the newly introduced feedback line is used to propagate the fault effect from the update flip-flop towards the shift flip-flop.

   (b) Next, the data is shifted through the shift path towards the scan output. During those two steps, the functional operation of an RSN is paused.

   (c) Finally, the data at the scan output of the scan segment is further propagated through an RSN by applying regular CSU operations.

2. **Testability Enhancement for the Reset Line**

   The reset lines of shadow flip-flops are testable with the help of the DfT enhancement. To perform a test, an RSN is set into a known state which differs from its reset state. A non-reset state is read from the shadow flip-flops into the scan flip-flops through the 0-branch of the feedback multiplexer, as shown in Fig. 5.2. This value is propagated towards the global scan-out by using conventional retargeting methods. Then, a global reset is applied to activate faults affecting the reset functionality. Next, the fault effects are read from the shadow registers through the feedback loop and shifted-out of the RSN. The shifted-out sequences for a reset and non-reset states are compared to detect a fault. Finally, a global reset signal is applied once again to bring the RSN into its initial state.

3. **Testability of the Feedback Loop Primitives**

   The faults affecting the additional feedback loop primitives are tested in multiple phases, while testing the D output of the instrument and hence the feedback multiplexer 0-input cannot be covered without controlling the instrument from outside. This paper considers faults within the RSN including the interfaces. Faults within the instruments lay out of the scope and do not contribute to the resulting coverage. In the first phase, the D-value is captured and observed by setting $FeedbackEn = 0$. Then, with $FeedbackEn = 1$, $\overline{D}$ is shifted into the loop and observed outside. If the feedback multiplexer output stayed still at D, the corresponding stuck-at-D faults at the multiplexer output, its 1-input or a stuck-at-0 fault at $FeedbackEn$ are detected. Next, D is shifted into the loop to detect stuck-at-$\overline{D}$ at the multiplexer output and 1-input. Finally, $\overline{D}$ is shifted again into the loop, and with $FeedbackEn = 0$ it will load D again, otherwise there is a stuck-at-1 fault at $FeedbackEn$.

5.1.4 Test of Control Primitives

This section presents a method to formally validate whether all the faults affecting the control primitives can be tested by observing an erroneously activated scan path with a changed length. The method below has been first published in [LWW21]. If at least one fault exists, which is not testable this way, the RSN is transformed into a testable functionally equivalent one with negligible hardware overhead. In the resulting RSN, it is guaranteed that all the single faults at the control primitives are testable and the existing methods to test RSNs can be efficiently applied to this RSN. First, we present a formal definition of the testability concept, then we present a scalable method for so-called series-parallel RSNs defined below, and show how an arbitrary RSN can modeled as a series-parallel one.

Testability Concept

In this subsection, we present the testability concept for control primitives of RSNs.

Definition 5.1 (Single fault reachable paths)**.** An active scan path asp_l is called to be "*single fault reachable*" from another path asp_k, if and only if there is a single fault f which activates the path asp_l instead of asp_k erroneously for some control input.

To check whether a given path is single fault reachable from another path, their activation conditions are compared, as shown in the example below.

Example: *In Fig. 5.3, a multiplexer* m_1 *has two inputs. The paths through the upper branch of the scan multiplexer are single fault reachable from the paths through the lower branch, by a single fault affecting the address control signal of* m_1*.*

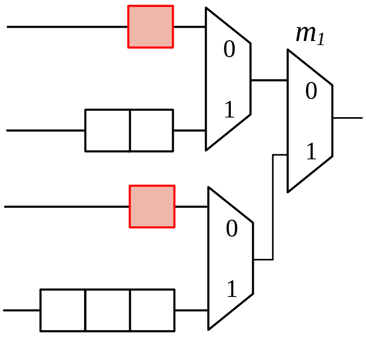


Figure 5.3: Testability concept example

If the paths arriving at different multiplexer inputs have different lengths, any fault of the multiplexer control can be detected.

Definition 5.2 (A fault detectable by an altered path length (DT-PL))**.** A single fault f in the RSN control logic logic is categorized as "*detectable by an altered path length (DT-PL)*", if under the same scan configuration, the length of the paths through a fault-free RSN is different compared to the length of any faulty path, which is single fault reachable from the initial path.

For a "detectable by an altered path length" fault, it is always possible to find a test sequence, which would detect the fault. In this case, for any path, which is erroneously activated in an RSN due to a single fault, the path length is different compared to the fault-free case.

The detection of a fault might be still possible, if there exist at least two paths through a fault location which have different lengths for a faulty and a fault-free case. However, the existence of equal path lengths leads to the case, where it is not sufficient to test the fault by activating any path. Instead, fault detection depends on the choice of an activated path and thereby is not guaranteed. In this case, a fault is categorized as a possibly "*undetectable by a path length (UDT-PL)*", since the existence of a test sequence is not guaranteed.

Example: *In Fig. 5.3, it is only necessary to compare the sets of lengths through the upper and the lower branches of* m_1 *to identify, whether the faults affecting* m_1 *are "detectable by a path length". If there exist at least one path length, which appears in both sets, it may not be possible to detect the fault affecting* m_1 *by an altered path length. In this example, the paths through the upper branch consist of* 1 *and* 2 *scan cells respectively. There also exist two other paths through the lower input of the multiplexer with the lengths* 1 *and* 3*. So, two paths shown in red have the same length, and the fault affecting* m_1 *is not "detectable by an altered path length".*

If an RSN contains any fault, which is not proven to be detectable by a path length, it is referred to as an untestable RSN. The goal of this section is not only to determine whether an RSN is testable, but also to pinpoint the exact single faults location in the control logic, which may not be *detectable by differences in a path length*, and to resolve such untestable spots via resynthesis.

Testability Analysis of Series-Parallel RSNs

A divide-and-conquer approach is formulated below, such that a series-parallel graph of an RSN is processed in a bottom-up manner. The analysis starts with elementary graph structures, such as parallel and series connections between the vertices.

For the vertices connected in parallel, the testability concept from Section 5.1.4 is applied, while the testability of the vertices connected in series does not depend from one another and thereby can be considered independently. As soon as the smaller subgraphs are processed, the analysis abstracts each such subgraph into a single edge and proceeds with the analysis of bigger subgraphs until the whole RSN graph is processed. The remainder of this subsection presents the details about the testability analysis implementation.

Let the set $asps(G_j)$ be the set of paths through a subgraph G_j, where each path asp_l has the length $pathLen_l$ and is activated if the path activation conditions $cond(asp_l)$ are satisfied. The set L_j contains all the path lengths through the subgraph.

For two subgraphs G_1 and G_2, the following cases are considered:

- **Series composition**:

For two serially-connected subgraphs, the set of path lengths through the resulting graph G includes all the possible combinations of the sums of the paths through the individual subgraphs.

$$\begin{aligned} L := \{pathLen_1{+}pathLen_2| \\ pathLen_1 \in L_1, pathLen_2 \in L_2\} \end{aligned} \tag{5.5}$$

At the same time, the conditions for activating the paths should not be contradicting:

$$cond(asp) := cond(asp_1) \wedge cond(asp_2) \tag{5.6}$$

A fault f in G_1, which is detectable by an altered path length, leads to a changed length of at least one path asp_1 through ‘:

$$pathLen_1^f := pathLen_1 + \lambda \tag{5.7}$$

where $pathLen_1^f$ is the length of a faulty path, λ is the relative change to the path length compared to the fault-free case, which arises due to a fault f.
Given the single fault assumption, the subgraph G_2 is fault-free, and any path through G, which includes an erroneously activated partial path through G_1, also differs from a fault-free path by the value of λ:

$$pathLen^f := [pathLen_1 + \lambda] + pathLen_2 \tag{5.8}$$

As a result, the fault f is detectable in G by an altered path length.

- **Parallel composition**:

For two subgraphs connected in parallel, the set of path lengths includes the lengths of the paths, which traverse one of the subgraphs:

$$L := L_1 \cup L_2 \tag{5.9}$$

The sets of path lengths should not be intersecting:

$$L_1 \cap L_2 = \emptyset \tag{5.10}$$

If the intersection is not empty, it indicates a problematic spot, meaning that the fault is undetectable by an altered path length. For all paths asp_1 and asp_2 through G, such that asp_2 is single fault reachable from asp_1 or vice versa, the information about the differences between the corresponding path lengths are saved.

The paths through an RSN are analyzed recursively with a help of a binary decomposition tree, and each time either a series or a parallel composition of subgraphs is considered. The initial computation starts with the left-most leave of the tree, and the vertices of the tree are traversed in the order of the reverse polish notation.

After the analysis is completed for the subgraph, it is abstracted to a single vertex of the binary decomposition tree, such that all possible path lengths through the subgraph are used for the annotation of this vertex. The computation continues with the next low-level subgraph, until all low-level graphs are processed, and then proceeds to a higher level, such that in the end the whole RSN is analyzed.

If all the target faults are detectable by a altered path length, then the RSN is already testable and the testability enhancing resynthesis is not needed for this RSN. Otherwise, the information about the control primitives with undetectable faults as well as the possible differences of the partial path lengths are used for resynthesis.

Example: *Given the decomposition tree from Fig. 4.4, the computation starts at the control segment cs_1. The tree is traversed following the reverse polish notation until the first parallel composition vertex is found. First, the subgraph consisting of the vertices P/m_1, s_1 and s_2 is analyzed, and possible path lengths are used for vertex annotation. The computation continues with analyzing the subgraph consisting of the vertices P/m_2, s_3 and s_4. As soon as all low-level subgraphs are analyzed, the higher-level subnetwork through the vertex P/m_{SIB} is analyzed.*

Testability-Enhancing Resynthesis

For the resynthesis of series-parallel RSNs, the following cases are considered for the subgraphs G_1 and G_2:

- *Series composition*: A fault in G_1 only affects the path lengths through this subgraph, and does not change the path length through the second subgraph G_2, and vice versa. Therefore, it is not required to consider the subgraphs simultaneously.
- *Parallel composition*: The lengths of the paths through the 0-input of the multiplexer m_i must be distinct from the lengths of the paths through the 1-input. Let $Diff = \{l_0 - l_1 | l_0, l_1$ path length through $0, 1$ input $\}$ and $m = min\{|c| | c \notin Diff\}$. If $m = 0$, the paths through the different multiplexer inputs are detectable by an altered path length. If $m \notin Diff$, we can insert m flip-flops in front of the 1-inputs of the multiplexers which leads to distinct path lengths. If $-m \notin Diff$, we can put m scan flip-flops in front of the 0-input, but not at the 1-input.

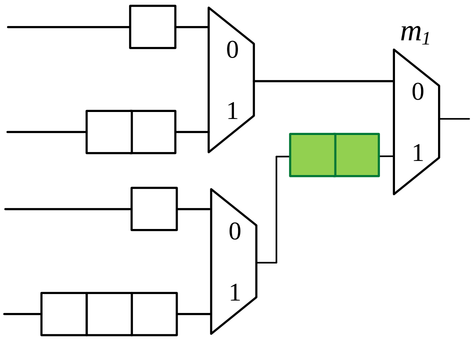


Figure 5.4: Resynthesis example: The testability issue from Fig. 5.3 is resolved by inserting two scan cells

Example: *Consider the example from Fig. 5.3 again. To ensure that a fault at m_1 is detected by an altered path length, two scan cells are added at the lower scan-input of the multiplexer m_i, as shown in Fig. 5.4.*

The resynthesis algorithm is applied recursively by traversing a binary decomposition tree in the same order as during the testability analysis. After the testability in a subgraph is enhanced, this subgraph is abstracted to a single vertex, which is annotated with the possible lengths of the path considering the newly added cells. After all the lower level subgraphs are processed, the computation goes one level higher in the binary decomposition tree, until the whole RSN is processed. In the resulting RSN, all the single faults in the control logic are detectable by a changed path length.

5.1.5 Test of Scan Segments

The method presented above ensures that the faults affecting the control primitives and the scan multiplexers are testable. In this section, the test of scan segments is considered, which has been first published in [WLA$^+$21]. In contrast to the existing schemes in Section 3.3, the test of scan segments is applied concurrently to an instrument access. A compact built-in self-test structure is added into the RSN and is used to generate a short test pre-sequence. This pre-sequence is augmented with a workload sequence, shifted to the tested RSN, and is used to check the shift logic of the scan segments in the currently configured scan path, as shown in Section 5.1.5. An example implementation of a concurrent BIST structure for RSNs is shown in Section 5.1.5.

Test Pattern Generation

Each complete test sequence (Fig. 5.5.a) includes a workload sequence W and a flush test sequence T. Flush test sequences are used to test the shift logic of the scan segments on the currently activated scan path. In general, a flush test sequence is symmetric with respect to inversion. This means that if a sequence $T =< t_{n-1}, ...t_0 >$ is a flush test for the activated scan path, then its inversion $\overline{T} =< \bar{t}_{n-1}, ...\bar{t}_0 >$ is one as well.

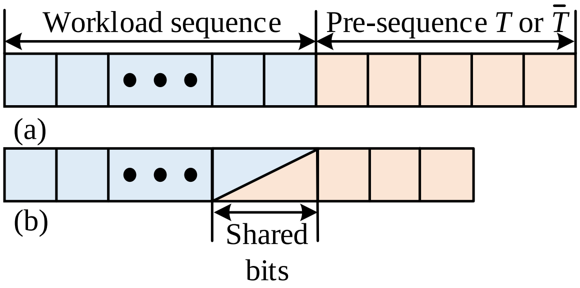


Figure 5.5: Test sequence construction a) workload sequence is augmented with a flush sequence b) bit sharing mechanism

For testing stuck-at faults in a scan path, the applicable sequences include a sequence "00110" and its inversion "11001". For different fault models, other flush test sequences can be used. The flush test sequences can be either provided by an automated test equipment (ATE) together with the workload sequences, or generated on-site, as discussed below in Section 5.1.5.

Fig. 5.5.b represents the bit sharing mechanism, which is used for merging the workload sequence with the flush pre-sequence. There, the last bit w_0 of the workload sequence $W =< w_{m-1}, ...w_0 >$ is used to decide, which of the tail flush test sequences (T or $\overline{T}$) overlaps with the head of W by at least one bit, and there is no need to repeat these overlapping bits in T or $\overline{T}$. For stuck-at-faults, the worst-case reduction in the test application time comprises 20% of a five-bit flush test sequence, if a constant overlap of the last bit is considered.

Test Pattern Application

ROSTI (RSN Online/Offline Self-Test Infrastructure) is a self-test structure for RSNs to generate test sequences and to attach them to the workload sequences. Its structure is shown in Fig. 5.6, and includes a test sequence generator (TSG), an acceptor and a controller. ROSTI is placed between the RSN and the TAP controller. Together with a TAP controller and an access port, ROSTI represents an access interface, which enables RSN self-test and is compliant with the P1687.1 standard proposal [CVTR20], which allows to use other access mechanism rather then just a JTAG TAP controller to access an RSN.

Data is propagated from a TAP controller through ROSTI to the RSN, and back towards the TAP controller. ROSTI operates as follows:

- After the capture signal and with the shift signal, a flush test sequence is generated in the test sequence generator, and it is inserted in front of the workload sequence.
- The flush sequence and the workload sequence are shifted towards the scan input of the RSN and are further propagated through the activated path.
- If the path is not corrupted, the bits of the pre-sequence are shifted-out unchanged, and the workload sequence is at the target instrument. If the path is faulty, the $Viol$ violation signal indicates a defect in the RSN.

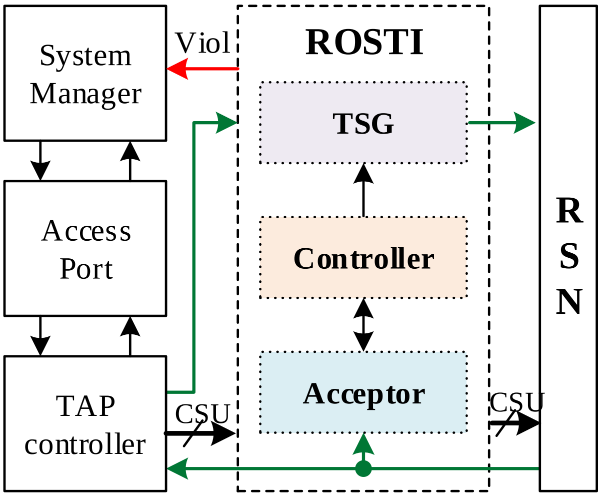


Figure 5.6: RSN Online/Offline Self-Test Infrastructure (ROSTI) structure

The idea behind ROSTI is valid for a wide range of fault models. To extend ROSTI for a fault-model of interest, the flush test sequence needs to be modified, as well as the exact implementation of the test sequence generation and acceptor blocks. ROSTI can be implemented as a simple hardware block as presented below. In the following, the hardware implementation is explained in a block-by-block manner including three parts:

1. *Test Sequence Generator (TSG)*

   The TSG is used to generate flush test sequences ("01100" or "10011" for stuck-at-faults) based on the first bit of a workload sequence, and to merge them together without adding any hold cycle. The TSG operates as follows:

   (a) *Reuse the first bit:* The first bit of the workload sequence is reused as the first bit of the generated flush test sequence.

   (b) *Generate and apply the flush test sequence:* The rest of the flush sequence are generated in a four-bit shift register.

   (c) *Apply the workload sequence:* As soon as the flush test sequence is generated and shifted into the RSN, the workload sequence starts to being shifted into the RSN for the whole length of the workload sequence.

2. *Acceptor*

   The acceptor is used to compare the shifted-out results with the expected ones, and to issue an internal violation signal if these values do not match. As soon as the workload sequence is shifted into the acceptor, its first bit is recorded into a flip-flop of the acceptor to distinguish which flush test sequence ("01100" or "10011") is used in a given test sequence. The acceptor is constructed as a finite state machine, which consists of few flip-flops and few logic gates. It is independent of the length of the ASP and its hardware costs depend only on the length of the test sequence since the acceptor is controlled by the available global shift and update signals.

3. *Controller*

   The major task of the controller is to generate the violation signal ($Viol$) with a correct timing. When a violation occurs, i.e. if an RSN test fails and the acceptor issues the internal violation signal, ROSTI raises the violation signal to the system. This signal is triggered by the rising transition of the clock signal after the removal of the shift signal and it holds for one cycle. The controller also is used to propagate the first bit of the workload test sequence from the test sequence generator towards the acceptor to allow correct test response comparison.

5.1.6 Test Integration

This section discusses integration of the presented DfT scheme into the RSN-under-test. First, a short summary of the necessary changes to the RSN structure are presented followed by some details about test sequence construction for the enhanced RSN.

Changes to the RSN structure

In Fig. 5.7, the example from Fig. 5.1 is enhanced with the required DfT changes, such that all the testability issues are resolved. The test integration can be summarized with the following steps:

1. **Control Primitives Testability Enhancement:** The RSN structure analyzed and it is checked whether any single fault in the control logic is undetectable by a changed

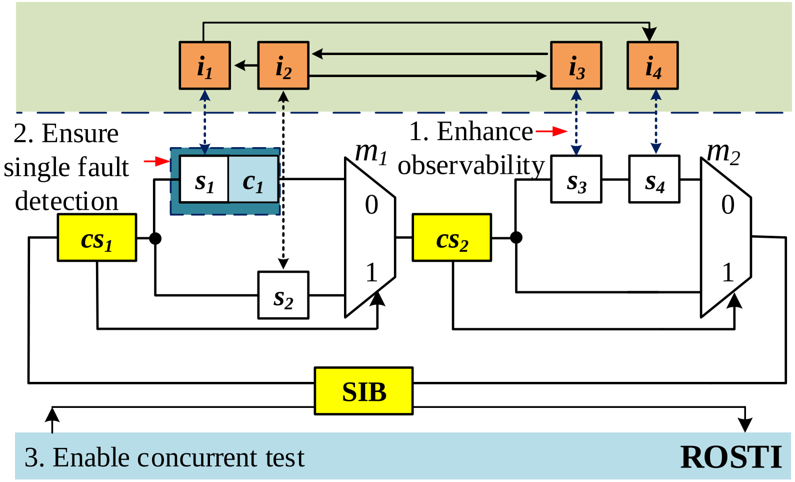


Figure 5.7: Testability-enhancement for the RSN from Fig. 2.1

path length. In this case, a minimized number of changes is applied to the RSN structure, so that the existing test methods can be applied to detect any single fault in the control logic by an altered path length. It implies, that if the path length is correct, then the correct path is activated through the RSN and a silent data corruption due to a fault in the control logic is excluded. Thereby it is ensured that the registers of the correct instruments are accessed. In our example, fault detection is ensured by adding a single scan cell c_1 before the multiplexer m_1.

2. **Scan Interface Observability Enhancement:** The design-for-test scheme is integrated to increase the observability of the shadow registers. As a result, the faults in the capture- and update-circuity of the scan segments become detectable, and the correct operation of the interfaces to the instruments (including the interface of s_3 in Fig. 5.1) can be tested. The reset functionality of the shadow flip-flops is now also testable.

3. **Scan Segment Test Enhancement:** The described compact BIST hardware is integrated into the RSN, which allows to test the shift logic of the scan segments concurrently. A fault affecting a scan segment is detected with the help of a flush test pre-sequence, as soon as a path through this segment is activated.

Test Sequence Construction

As soon as the testability flaws in the initial RSNs are identified and resolved, a sequence of efficient test patterns can be generated and applied to the RSNs. To test specific scan segments, it is required to include them into an activated scan path. A test sequence set can be generated automatically to cover the whole RSN structure with a minimized test application time, as in [BKW15b, CSSS18, HHD21].

Each complete test sequence includes an instrument test sequence W, which is used for testing the interface to instruments, and a flush test sequence T, which is responsible for testing the shift logic of the scan segments on the currently activated scan path. After the test is applied, in a fault-free case, the flush test sequence will be shifted-out unchanged, and the bits of the workload sequence would contain the test results for the scan interfaces. The length of the shifted-out sequence is used as an indicator for the single faults in control logic. The same applies also for the single flip-flop transparency faults in the shift registers, since they reduce the length of the activated path by one shift cycle.

5.2 Extension for Online Periodic Test of RSNs

The DfT method presented below significantly improves the testability of RSNs, and faults can be detected offline and online. Offline test of RSNs is thoroughly investigated [UKW17, KBSW16, CDRS20, CSSS18, HHD20], and online concurrent test method for RSNs has been presented in [WLA$^+$21]. However, to avoid fault accumulation in rarely used components of RSNs, e.g. due to aging [LSW20], an online periodic test method is essential.

This section discusses the first online periodic test method for RSNs, which has been presented in [LWW22b] and complements the previously described DfT scheme. The developed algorithm generates a short sequence of test patterns, which tests all parts of an RSN. The generated sequence is uploaded on-chip and is applied periodically to avoid fault accumulation in RSNs. The overall test application time is minimized to comply with the timing requirements. The generated test sequences can be reused throughout the whole RSN lifetime, e.g., for offline structural test.

The remainder of this section is organized as follows. First, Section 5.2.1 provides a top-down overview of the presented test generation method. In Section 5.2.3, it is shown how to generate a minimized set of active scan paths, which cover all the components of an RSN. In Section 5.2.2, a scheduling approach is provided, which determines the order of the path activations. Thanks to scalable processing method, it is not required to exhaustively consider all possible transitions between the selected paths.

5.2.1 Test Generation Method Overview

Due to stringent real-time operation and performance requirements, periodic test must be performed within a limited time frame. For the major time, the system must operate in a functional mode, as shown in Fig. 5.8. If the time budget is not sufficient for executing the complete test, the test is usually divided into multiple sessions, each within the budget, and applied successively [KMM^{+}21, EZ17].

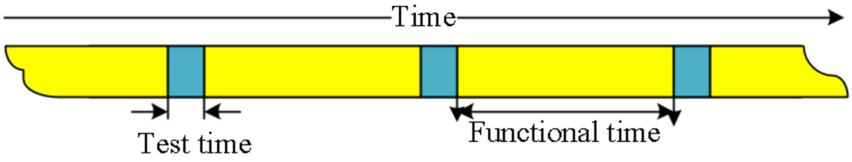


Figure 5.8: Periodic test

The test method presented below generates a short set of test access sequences, which are stored on-chip and are applied to the RSN periodically. Through a set of sequences, the RSN is configured in a way that all scan primitives, which include the scan segments and the inputs of scan multiplexers, are covered within a minimized test time. Thereby, the control signals are also implicitly covered. The test workload sequences can be generated by the existing methods [UKW17, KBSW16, CDRS20, CSSS18, HHD20]. The remainder of this section formulates the test sequence generation problem in terms of graphs and presents a top-down overview of the solution.

Problem Formulation

The solution to the scheduling problem is a minimized set of test access sequences $Test := \{Seq_0, \dots Seq_k \dots Seq_N\}$ for $k = 1, \dots N$, such that all the vertices of the RSN graph are covered at least once and the overall test application time is minimized. The following constraints must hold:

- Each vertex is covered at least once:

$$\bigcup_{Seq_j \in Test} \bigcup_{asp_i^j \in Seq_j} V(asp_i^j) = V \tag{5.11}$$

- The cost of the test sequence set is minimized:

$$\sum_{Seq_j \in Test} \sum_{asp_i^j \in Seq_j} cost(asp_i^j, Seq_j) \rightarrow min \tag{5.12}$$

  where $cost(asp_i^j, Seq_j)$ is the cost of adding a path asp_i^j at the end of a sequence Seq_j.

  For a sequence Seq_j and a path asp_i^j, the cost is calculated as follows:

$$cost(asp_i^j, Seq_j) := switch(asp_i^j, Seq_j) + 2 + |asp_i^j| \tag{5.13}$$

  where $switch(asp_i^j, Seq_j)$ represents the number of cycles which are necessary to configure the path asp_j from the last path in the sequence Seq_j; 2 cycles are required to perform capture and update phases, and $|asp_i^j|$ cycles are required to perform shift operation when the path asp_i^j is configured.

Top-Down Overview

For the problem above, it is possible to obtain an optimal solution, if the computing runtime and the storage capacities are not limited. However, in this case, all possible transitions between the scan configurations must be compared exhaustively, which is not feasible even for medium-sized RSNs. To efficiently generate the test sequence set, an efficient heuristic is presented:

- If all possible transitions between the scan configurations would be considered which are reachable by applying an unlimited number of CSU-operations, an optimum solution would be obtained. However, it is infeasible to explore the resulting solution space even for medium-sized RSNs. To reduce the solution space, at each step, we only consider such pairs of scan configurations (c_1, c_2) that the second configuration c_2 is reachable from the first configuration c_1 by applying a given limited number of CSU-operations.

- In an RSN, the number of configurations may grow exponentially in terms of the number of configuration bits. Therefore, exploiting all possible transitions between the scan configurations is infeasible, even if the number of considered CSU-operations is restricted. In Section 5.2.3, it is shown how to select those active scan paths, whose activation would bring the highest additional gain among all other candidates with respect to the coverage, without exhaustively checking all the candidates.

5.2.2 Scheduling of Test Accesses

Vertex Covering Problem Formulation

A test access scheduling method identifies a set of active scan paths, which cover all the reachable scan primitives in the RSN, and also their activation order. To comply with safety requirements, the test sequence set is minimized with respect to the overall test application time. Let the set ASP include all active scan paths asp_i^j, which are included into the test sequence set at a given time.

The algorithm is applied to the RSN graph, where each vertex is annotated with two values:

- $cost(v_j, asp_i^j)$ shows the additional cost of including the vertex v_j into a path asp_i^j and is defined as the length of the corresponding scan primitive. The costs of all the included vertices in the path asp_i^j are summed up to calculate the length of the path asp_i^j.

- $gain(v_j, ASP)$ represents the gain of covering the vertex v_j. The gain equals to 1, if the vertex has not been previously included into any active scan path in ASP. After a vertex is included at least into one active scan path, its gain is reset to 0.

The transitions between the activated configurations are represented by using the transition relation. Since storing the complete transition relation is impractical even for medium-sized RSNs, only rather small parts of the transition relation are generated dynamically and stored in the local memory. The transition relation for the running example is shown in Fig. 5.9. Each bit in a configuration corresponds to a scan multiplexer. The bit is set to 0, if the 0-branch of the multiplexer is selected and to 1 otherwise. Each path can be reset to the initial path asp_0 within 1 CSU-operation, and the reset transitions are omitted in the figure.

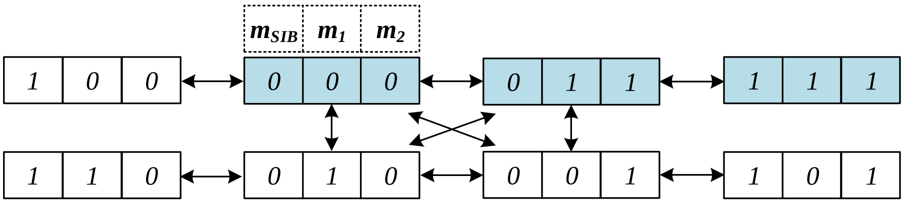


Figure 5.9: Possible configurations for Fig. 2.1

Each vertex of the transition relation graph corresponds to a single path. The edges of the graph show the reachability between the corresponding vertices within one CSU-operation.

At each iteration it of the algorithm, an intermediate test sequence set T_{it} is updated and some new paths are added. At each point, it is possible to assess the cost and the gain of adding a path asp_i^j into the sequence set T_{it}:

- The cost of adding a path asp_i^j at the end of a sequence Seq_j is referred to as $cost(asp_i^j, Seq_j)$ and is defined as in Eq. 5.13.
- The gain of adding asp_i^j into the sequence set T_{it} is referred to as $gain(asp_i^j, T_{it})$. It is defined as a number of vertices in the RSN graph, which have not been covered by the set T_{it}, but are covered by the path asp_i^j.

Test Sequence Generation

Algorithm 5.1 presents the general scheduling flow:

- *(Line 1-5)*: The computation starts with initializing the initial test sequence set T_0 with the first sequence Seq_0 and adding the ASP asp_0 into the sequence Seq_0.

Algorithm 1: *generateTestSequenceSet*

```
Input: RSN graph $G := (V, E)$, where the initial path $asp_0$ is activated
Output: Test as a set of sequences $Test := \{Seq_0, \dots Seq_k\}$, where $Seq_j := \{asp^j_0, asp^j_1, \dots asp^j_i\}$; Boolean flag
        $CoveredStatus$ which shows, whether all the requested vertices are covered
/* Initialize the initial state $asp_0$; the first path sequence $Seq_0$; and the test set $T$ */
$asp_0 \leftarrow reset$;
$Seq_0 \leftarrow (asp_0)$
$T_0 \leftarrow \{Seq_0\}$
/* Initialize the covered vertices with the vertices of $asp_0$ */
$V_{cov} \leftarrow V(asp_0)$
/* $asp_0$ is included into each path $Seq_j$ so it is not needed to cover it explicitly */
$V(asp_0).resetGain()$
/* Initialize the distance value */
$maxDistance \leftarrow m$
/* Initialize the iterator */
$it \leftarrow 0$
/* Proceed until all the vertices are covered */
while $(V_{cov} \neq V)$ do
    $k \leftarrow k + 1$
    /* For each sequence in the test set */
    for $Seq_j \in T_{it}$ do
        /* Find the candidate paths with a minimum distance from $Seq_j$ and relevant positive gain
           */
        $candidates \leftarrow getPathsMaxGainMinCost(Seq_j)$
        /* Select the path to add: if multiple branch-and-bound */
        $addedPath(Seq_j) \leftarrow selectPath(candidates)$
    end
    /* Collect the potentially added paths */
    $AddedPaths \leftarrow \bigcup_{Seq_j}^{Test} addedPath_j$
    /* If it is not possible to find at least one path within the specified distance from any of
       the sequences then stop the operation */
    if $AddedPaths == \emptyset$ then
        $break$
    end
    /* Select such a sequence $S$ in $T_{it}$ and a path $asp$, which have the shortest distance between
       them */
    $(S, asp) \leftarrow getTheSequenceToAppend(T_{it}, AddedPaths)$
    /* Add the path at the end of the selected sequence */
    $S' \leftarrow S + asp$
    /* Update the coverage */
    $V_{cov} \leftarrow V_{cov} \cup V(asp)$
    /* Reset the gain of the included vertices */
    $V(asp).resetGain()$
    /* Update the test set with the selected sequence and ensure that the reset sequence is
       still in the test set */
    $T_{it+1} \leftarrow T_{it} \setminus \{S\} \cup \{S'\} \cup \{Seq_0\}$
    $it \leftarrow it + 1$
end
/* After all the sequences are generated, return the final test set and the coverage status */
$Test \leftarrow T_{it}$
return $(Test, V_{cov})$
```

  - *(Line 6)*: The maximum number of CSU-operations for test sequence generation is initialized. Depending on the selected value, the trade-off between the runtime and the test application time is established.

- *(Line 8-24)*: The test sequence generation runs until either all the vertices of the RSN graph are visited at least once or it is not possible to cover any more vertices (Line 16).

- *(Line 10-13)*: For each sequence in T_{it}, possible paths are determined, which are reachable within a specified number of CSU-operations, and which allow to cover more vertices in the RSN graph ($gain(asp_i^j, T_{it}) > 0$). If multiple such paths exist, the added path is selected by using a branch-and-bound approach.

- *(Line 14-17)*: If none of the sequences in the test set can be extended by a path, the test generation converges.

- *(Line 18-21)*: The sequence Seq_j to augment with an additional path asp_i^j is selected, such that the cost of adding the path into the sequence is minimized ($cost(asp_i^j, Seq_j) \rightarrow min$).

- *(Line 22)*: The test sequence set is updated. The selected sequence is augmented with the selected path. It is explicitly ensured that a basic sequence, which only includes the reset ASP, is still in the set.

- *(Line 26)*: The algorithm provides the test sequence set $Test$ and the covered vertices as an output.

Example: *In Fig. 5.9, one CSU-operation is considered at a time ($m = 1$). The computation starts at the reset configuration $\{m_{SIB}, m_1, m_2\} = \{0, 0, 0\}$. The configurations $\{1, 0, 0\}, \{0, 1, 0\}, \{0, 0, 1\}$ and $\{0, 1, 1\}$ are reachable. The configuration $\{0, 1, 1\}$ has the highest gain and is added into the sequence. Next, it is possible to add a configuration $\{1, 1, 1\}$ into the existing sequence, or to initialize a new sequence, which starts with a reset configuration and includes also one the reachable configurations. The configuration $\{1, 1, 1\}$ has the highest gain and is selected for adding into the existing sequence. All the vertices of the RSN graph are covered with three configurations, which belong to the same sequence.*

5.2.3 Selection of Relevant Active Scan Paths

From a given path, an exponential number of other paths is reachable within one CSU operation, as illustrated in Fig. 5.10. Therefore, comparing all the paths, which are reachable within the threshold is not feasible to identify the relevant paths.

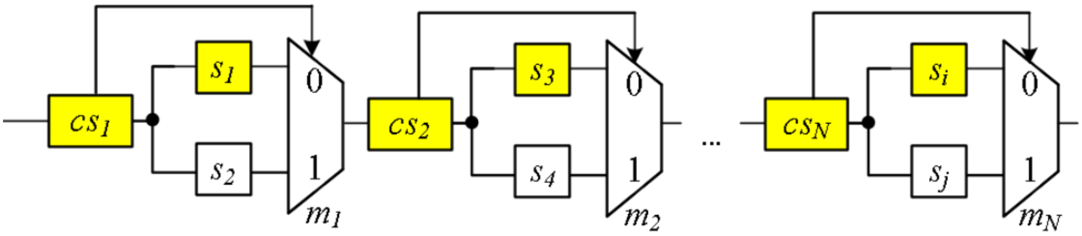


Figure 5.10: RSN chain: Each scan multiplexer is controlled independently

This section provides a method to quickly identify those paths, which might have the highest gain among all other candidates. First, an path selection method is presented for series-parallel RSNs. Then we show, how the presented method is applied even for those RSNs, which are do not have a series-parallel property.

5.2.4 SP-RSN Path Selection

Given an initial path asp_i^j, it is not required to consider all the paths, which are reachable from this path within a given number of CSU-operations. Instead, a relevant subset of paths is generated for asp_i^j by mutating the corresponding scan configuration within a given number of CSU-operations.

As discussed in Section 4, a series-parallel RSN graph can be represented by its binary decomposition tree. A binary decomposition tree for the example from Fig. 2.1 is shown in Fig. 5.11. The initially activated scan path is highlighted in yellow. Identifying a valid path in a tree has a logarithmic complexity. The tree traversal starts from the top vertex and continues in a depth-first-search manner. First, only the top-level vertex is selected for a path. At each step, the algorithm tries to select more vertices. The following rules are applied. When the algorithm traverses a "P" vertex, only one arbitrary child vertex is selected. If an "S" vertex is met, both children are selected. The computation continues until it is not possible to include more vertices.

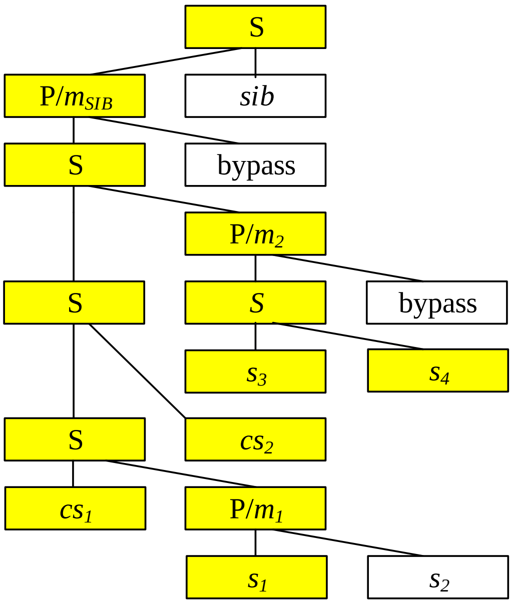


Figure 5.11: Active Scan Path in a Binary Decomposition Tree from Fig. 4.4

For a given scan configuration, it is possible to identify those other scan configurations which are reachable from a given configuration within 1 CSU operation and which have a high gain. In a binary decomposition tree, a path is activated, and a corresponding scan configuration is computed. We traverse the vertices of the tree in reverse Polish order. The multiplexers at the lowest levels of the binary decomposition tree are selected. The developed algorithm flips the states of selected scan multiplexers compared to the initial scan configuration. If more CSU operations are considered, the flipping is also allowed for the second lower-level multiplexer vertices.

Example: *In Fig. 5.11, the initial path is shown with yellow. The next path is obtained by flipping the states of the vertices* m_1 *and* m_2*. The next path traverses the lower branches of* m_1 *and* m_2 *through the vertices* s_2 *and* s_4.

5.2.5 Non-SP-RSN Path Selection

Consider the non-series-parallel RSN graph in Fig. 4.7 and its series-parallel representation in Fig. 4.9. The newly added vertex s_{2c} in Fig. 4.9) is referred to as a twin of the initial vertex s_2. For the means of testing, the twins are mutually equivalent. If one of the twins is tested by a given sequence then another one is tested as well. Applying the procedure above for series-parallel representation of non-series-parallel RSNs results in a pessimistic estimation of the coverage. The resulting test pattern sequence, which is generated for a series-parallel representation, covers all the primitives in the original RSN but may require a longer test application time. A longer test application time may be needed if the initial vertex is tested more than one time in the resulting test sequence set due to the RSN structure.

5.3 Evaluation

This section discusses the experimental results for the developed test and testability-enhancing methods. The detailed experimental data for this chapter is provided in Appendix B.2. The remainder of this section is organized as follows:

- Section 5.3.1 discusses the experimental results for the developed DfT scheme.
- Section 5.3.2 provides the results for the presented online periodic test scheme.

5.3.1 Design-for-Test Scheme

The complete design-for-test method is implemented and evaluated on a wide range of benchmarks. It uses Instrument Connectivity Language (ICL) descriptions of RSNs as an input for test generation and generates HDL descriptions for gate-level synthesis.

Scan Interfaces

A gate-level description of a scan segment is enhanced. A feedback line is injected to improve the testability of a scan interface and a reset line. Enhanced scan segments are used further as scan primitives for all RSN benchmarks during test sequence generation and synthesis.

Control Primitives

The developed DfT method ensures that the RSN is testable with respect to single faults in the control logic. If all faults in the control logic are detectable by an altered path length, the testability of the RSN is algorithmically proven. The ability to prove this property for any arbitrary RSN structure eliminates the danger of silent data corruption with respect to single faults in the control logic and is thereby one of the major contributions of this chapter. To ensure fault detection, the lengths of a minor number of scan chains may be slightly increased.

Original Benchmarks

The scalability and the effectiveness of the developed method has been proven using the benchmarks from the ITC'2016 [TJD$^+$16], DATE'2019 [RTB$^+$19] and ITC'2002-based [BKW15b] sets. As shown in Table 5.1, for the benchmark designs *TreeBalanced, Mingle, BasicSCB,* the presented testability analysis approach identified single faults in the switch logic, which are undetectable by an altered path length. A minor number of scan cells, which ranges between 4 cells for *Mingle* and 8 cells for *BasicSCB*, has been added into the initial RSN structure to ensure fault detection by the existing methods. A negligible runtime (below 3 seconds) has been required to perform both the analysis and the resynthesis.

	Faults	Undetected Faults	Scan Cells	Added Scan Cells	Runtime [s]
BasicSCB	40	8	176	4	1.0
Mingle	52	16	270	8	1.2
TreeBalanced	200	12	5581	6	2.1

Table 5.1: Added cells for enhancing the testability of the original benchmarks

As expected, for a major part of all widely-approved benchmarks, each fault in the RSN switch logic is detectable by an altered path length. Using the presented approach, it has been algorithmically proven that the remaining benchmarks are testable with respect to single faults.

Artificial Benchmarks

Since for most initial benchmarks, the testability property has been proven, the applicability of the structural resynthesis up to this point has been only validated on a limited number of benchmarks. For the third-party RSN designs the testability property is not guaranteed, and the scalability and efficiency of the resynthesis must be validated as well. To create a representative benchmark set, which contains a high number of large RSN designs, additional bypass registers with a controllable length are implemented. The testability of the modified benchmarks has been analyzed and the benchmarks have been resynthesized to ensure the unique path lengths within a single fault assumption in the switch logic. Fig. 5.12 shows the ratio between the number of cell, which have been added into an artificial RSN benchmark to enhance its testability, and the total number of scan cells in this RSN.

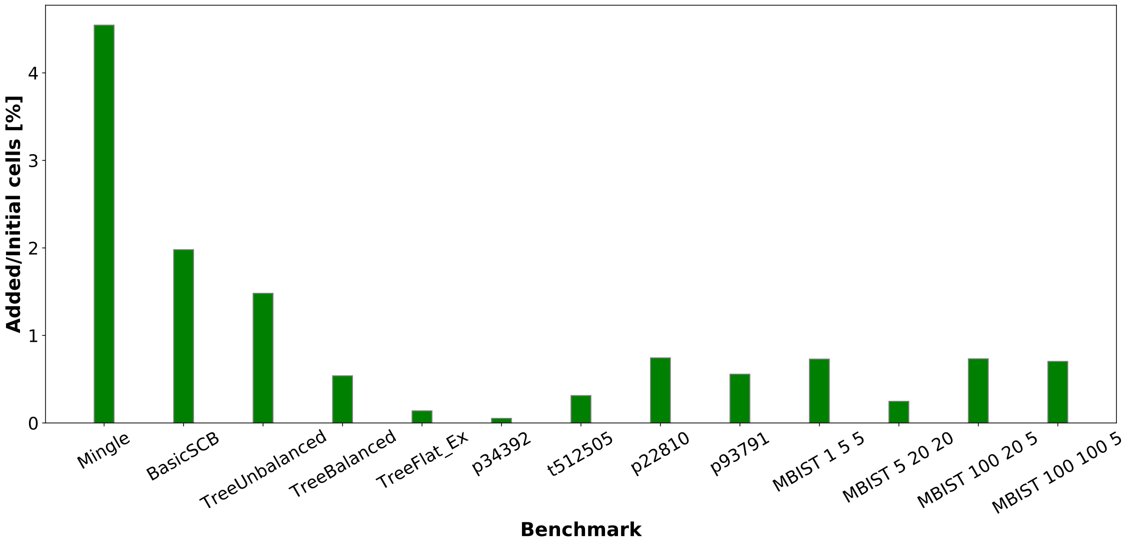


Figure 5.12: Added cells for enhancing the testability of the artificial benchmarks

The computed ratio shows that the hardware overhead of the DfT enhancement is low for all the considered benchmarks. As the size and the complexity of RSNs increase, it becomes negligible. The runtime of the approach is acceptable and requires less than 25 minutes for the largest benchmarks, and just few seconds for the most of the benchmarks. Thereby, the presented method is scalable for large and complex RSN designs.

Scan Segments

To test scan segments, an RTL description of ROSTI has been developed. ROSTI requires four flip-flops for the test sequence generator, and another four bits for the acceptor. For the ROSTI controller, eight flip-flops are used. The architecture of ROSTI is independent of the RSN and the number of the required flip-flops is also fixed for any RSN under test. The delay overhead of ROSTI is negligible and is less than 0.07 ns. To test ROSTI itself, a commercial tool has been used to perform test pattern generation with full-sequential ATPG setting. It achieves a fault coverage of 96.84% with 16 patterns and 278 test cycles. The developed DfT enhancements for scan interfaces and scan segments are independent from the RSN. The gate-level fault coverage for stuck-at-faults is determined with a commercial sequential stuck-at fault simulator.

Test Integration

In this section, the complete developed DfT approach is evaluated. To evaluate the testability-enhancing resynthesis for control primitives on a wider benchmark set, while being able to assess the DfT enhancements for scan interfaces and a shift logic, the RSN designs have been constructed based on the ITC'02 SoC benchmark set [MIC02].

In Fig. 5.13, the fault coverage for RSN benchmarks without feedback lines in the scan segments is compared to the fault coverage obtained when the complete DfT scheme is integrated. Fault coverage is 94.72% on average. To mitigate the coverage gap above, it is necessary to test the interfaces to instruments and logic. If scan segments are enhanced by integrating a feedback line and the workload patterns are used to test scan interface, the complete fault coverage is obtained for all the benchmarks.

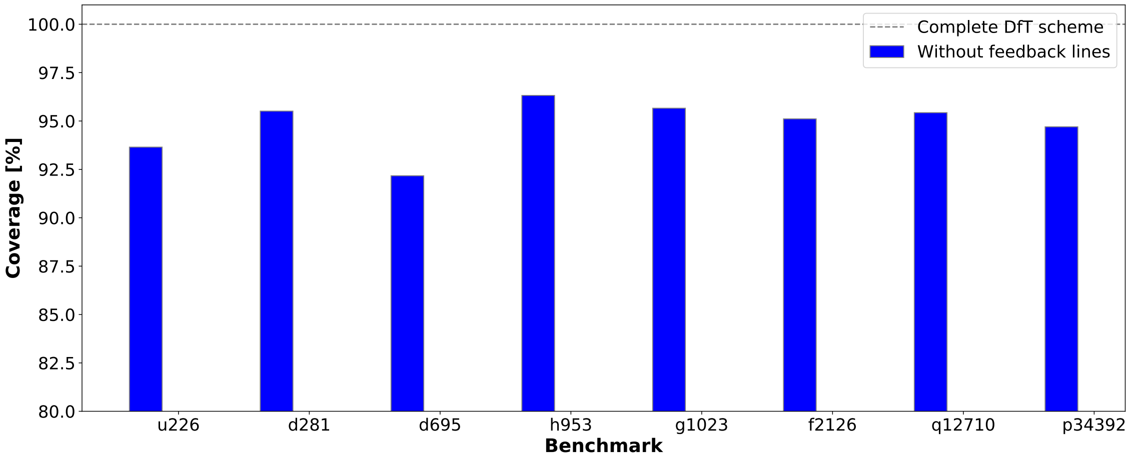


Figure 5.13: Fault coverage comparison

In the resulting RSNs, faults in scan interfaces, control primitives and scan segments are detectable, and the test integration scheme is applied to reduce the test cost. The test sequences of the scan interfaces are now integrated into the workload, and a flush sequence tests the shift logic. The length of an activated path is used as an indicator for faults in the control logic. Fig. 5.14 shows the test cost reduction for the case when the developed test integration scheme is considered and the bit-sharing mechanism is activated compared to the case when the test sequences for scan interfaces and scan segments are applied individually one after another. For all the benchmarks, the test integration scheme provides a significant test cost reduction.

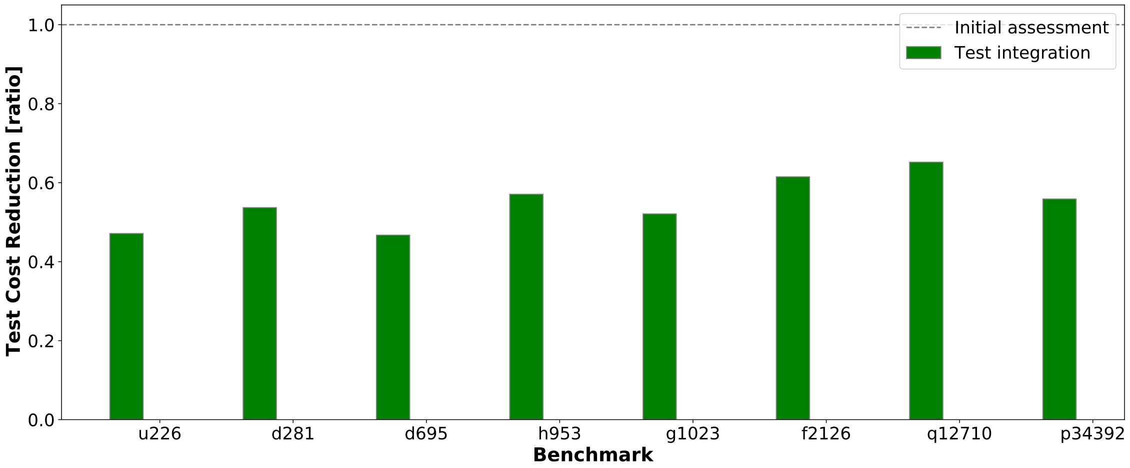


Figure 5.14: Test cost reduction by test integration

The scalability and effectiveness of the developed DfT enhancements has been shown for a wide range of benchmarks. The absolute numbers for the test cost are given in Section B.2.

5.3.2 Online Periodic Test

Test access sequence sets for performing online periodic test of RSNs have been generated according to the developed method. The coverage of scan primitives is defined as a fraction of scan primitives, which are accessed by a given test sequence set, from the whole set of scan primitives. The solution space has been explored and a tradeoff between the test cost and the coverage of scan primitives has been investigated. A set of close to pareto-optimal solutions has been generated by means of genetic algorithms. For a required coverage, the best generated solution with respect to test cost has been selected.

Full Coverage

First, the results for the test sequence sets with full coverage of scan primitives are presented for all the benchmarks. Fig. 5.15 shows the overall test cost in terms of the number of activated scan paths. For all the benchmarks, it has been possible to cover all the scan primitives with a rather low number of paths.

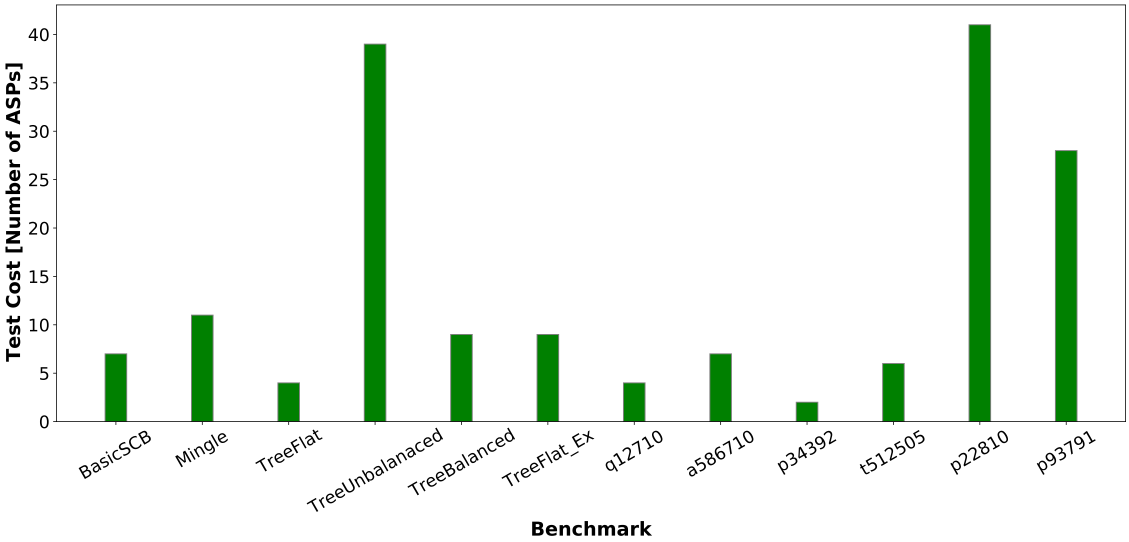


Figure 5.15: Test cost, number of activated paths (ASPs)

Fig. 5.16 shows the corresponding number of independent test sequences. For larger benchmarks (such as *TreeUnbalanced, p22810, p93791*), which require activating more scan paths to achieve full coverage, these test sequences can be applied individually during independent online periodic test sessions to meet safety margins.

Exploring the Tradeoff

Using a genetic algorithm, it is possible to explore the tradeoff between the required coverage and the resulting test cost. To evaluate this idea, the test sets with a requirement of 90% coverage are generated additionally to the ones with a full coverage. Fig. 5.17 shows the ratio between the test cost for the 90% coverage requirement and the original test cost. For all the benchmarks, by relaxing the coverage conditions by just 10%, the test cost can be significantly reduced.

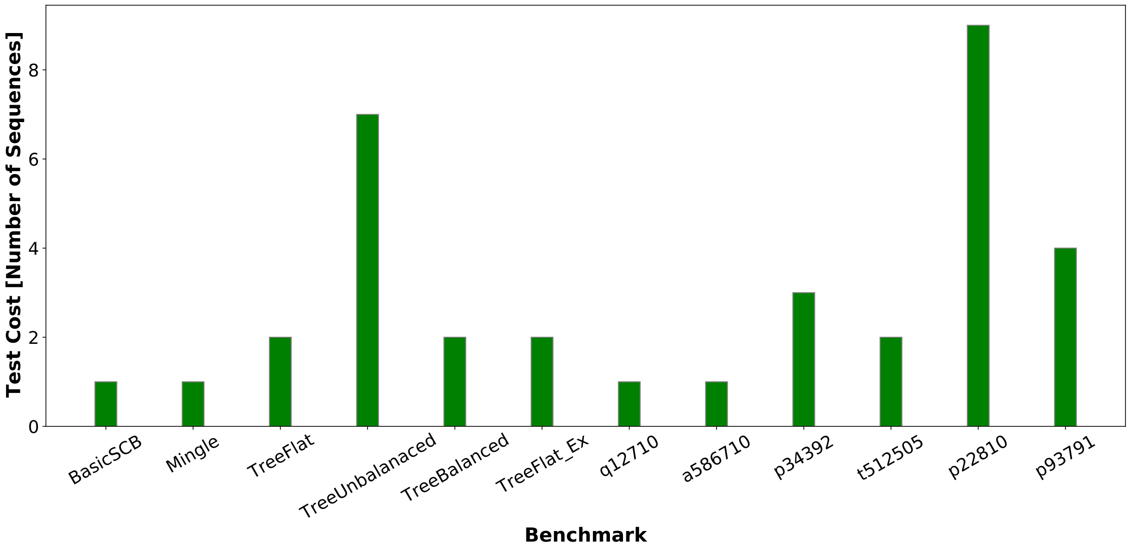


Figure 5.16: Test cost, number of sequences

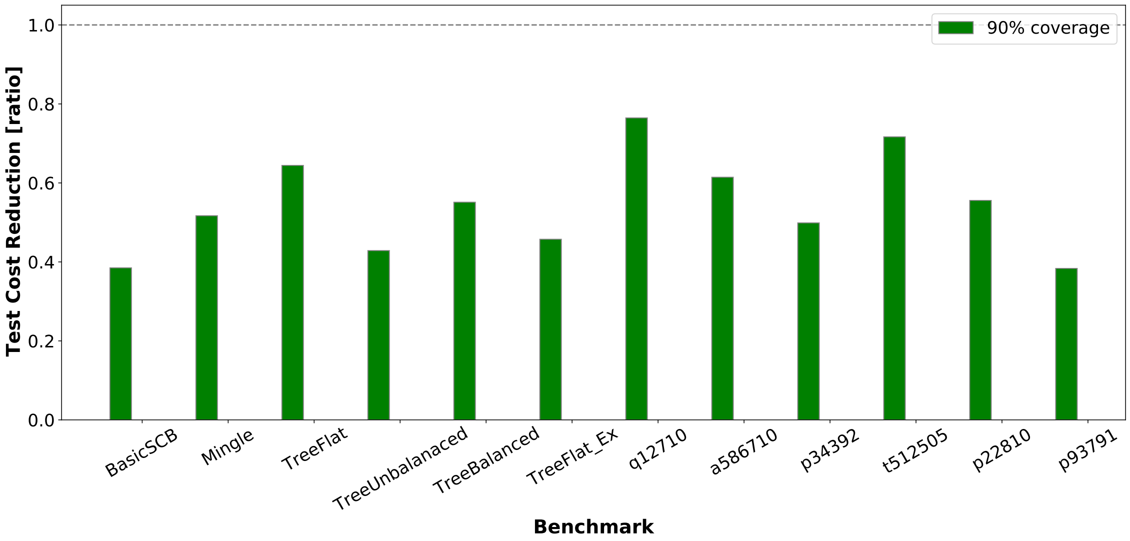


Figure 5.17: Test cost reduction for 90% coverage compared to full coverage, cycles

The experimental results show that the developed method is efficient for a wide range of RSN designs and is scalable with an increasing size of RSNs. Scalable graph-based modeling and efficient mapping to a genetic programming problem instance allow to generate a test sequence set with a required coverage, while minimizing the test cost. The detailed experimental results, including the absolute numbers, are given in Appendix B.2.

5.4 Summary

Reconfigurable Scan Networks are widely used for accessing instruments during debug, test and validation, as well as for performing system-level-test and online system health monitoring. Even if the most critical parts of RSNs are hardened, the probability of a fault within an RSN cannot be neglected. The correct operation of RSNs is essential, and RSNs have to be thoroughly tested. The existing test methods have limitations with respect to fault detection due to low observability and controlability of certain RSN primitives and cannot be readily applied to perform online periodic test. Moreover, due to inherently sequential structure and complex control dependencies of RSNs, testability analysis and enhancement of RSNs is an extremelly challenging problem.

This chapter presents a method to systematically enhance the testability of RSNs with the help of the complete design-for test scheme. The developed scheme significantly enhances the RSN testability, such that faults affecting the interfaces to the instruments, the control primitives and the scan segments can be tested. Each test sequence may cover multiple faults, which allows to significantly optimize the size of the test sequence set. The presented scheme is flexible with respect to the fault model, has a low hardware overhead and does not require changing the RSN topology rules. Therefore, it is compliant with the existing test and diagnosis methods for RSNs and is supposed to be used complementary to these schemes. The scheme is also flexible with respect to an access mechanism, and can be controlled by the workload test patterns from an ATE, from the cloud or even stored on-chip internally. The presented scheme generates test sequences with complete fault coverage and reduced test cost. If an RSN is equipped with the developed DfT scheme and the existing test methods are applied, defect RSNs can be efficiently ruled out during the manufacturing test.

Due to aging, testing an RSN once is not enough to ensure reliable access via RSNs throughout the lifetime. Even if scan segments are tested concurrently to the functional operation to check the shifting logic of the currently activated scan segments, rarely accessed components might still accumulate faults. To avoid this, an online concurrent test is complemented with an online periodic test. The developed online test method examines all scan primitives within a safety interval with a minimized number of test patterns. The generated test sequences are used throughout the whole RSN lifetime, both as an online periodic test and as an efficient offline manufacturing test with a reduced test application time on ATE.

Chapter 6

Security Compliant RSNs

Modern systems integrate an increasing number of IP-components, which may come from different vendors. Some of these components may contain confidential information, which should not be disclosed to an unauthorized user. Some other components might come from an untrustworthy supplier, which might want to sniff this information. To prevent unauthorized access and information leakage, the connections between the components in the original system are carefully designed. These connectivities together with an explicit security specification are referred to as the security properties of a system.

Accessing RSNs may introduce additional connectivities into the original system, if the RSN is integrated improperly. These connectivities can be misused as side-channels to leak or manipulate sensitive data. However, it cannot not be a responsibility of a design-for-test engineer to repeat a complete security analysis of the original system. Instead, an automated solution must be able to identify all unwanted extra-connectivities due to improper RSN integration and efficiently resolve them. The existing solutions do not provide a systematic approach to validate the compliance of a given RSN with the security properties of the underlying system and are not able to identify all the violating connectivities.

The violating connectivities due to improper RSN integration must be precluded either functionally, by restricting the set of test sequences using filters [BKW14, AKS+18], or structurally, by resynthesizing some parts of the RSN [RKA+18, RTB+19]. The existing methods either have unwanted side-effects and also block the authorized access to instruments [BKW14, AKS+18], or resolve the violating connectivities locally, consider one violation at a time, and thereby incur many structural changes [BKW14, AKS+18].

The remainder of this chapter is organized as follows:

- In Section 6.1, a security compliance analysis is presented, which identifies all security compliance violations due to improper RSN integration. The developed approach has first been published in [LAR$^+$19] and then enhanced in [LWW22d].
- In Section 6.2, a security-preserving resynthesis is presented to prevent information leakage due to improper RSN integration. The developed approaches, which have been first published in [LAWW20, LWW22d, LAW21], overcome the limitations of the existing methods and integrate security compliant RSNs for a single user and also for multiple users with different access rights.

6.1 Security Compliance Analysis of RSNs

This section presents a security compliance analysis of RSN. The remainder of this section is organized as follows:

- The security properties of the initial system are extracted from the design description and from the explicit security specification as shown in Section 6.1.1.
- A precise reachability analysis of the RSN is performed, as detailed in Section 6.1.2.
- The reachability properties of the resulting system, which consists of the original system and the integrated RSN, are computed as shown in Section 6.1.3.
- Finally, the security compliance of the RSN with the given system is verified as shown in Section 6.1.4. All the security compliance violations introduced into the system due to the RSN integration are identified.

6.1.1 Security Specifications

Integrity against unauthorized accesses is an essential property of a dependable RSN. The connectivities through the RSN must be compliant with the security properties of the system, as discussed in this section.

The security properties can be defined as implicit and external security specifications by a system designer:

- *Implicit specifications* follow from the design. If a physical connection or a path along which two instruments or components could communicate does not exist, such a connection must not be introduced by the RSN infrastructure. An implicit security specification of the system is defined by the reachability properties of the vertices of the initial instruments graph $G^I := (V^I, E^I)$. Each edge $e := (i_1, i_2) \in E^I$ of the graph reflects a connectivity from the instrument i_1 to the instrument i_2 through the initial functional system. The structural reachability of the instruments graph can be computed by using Floyd-Warshall's algorithm [War62]. To identify the functional connectivities inside the system, false path analysis [HP13, NTKW98, NSV$^+$17] or SAT-based methods [SRS$^+$16] can be applied.

- *Explicit specifications* can be formulated by the designer either to allow specific connections through the RSN, which are not present in the original design, or to exclude connections even if they were physically present in the design. The latter may be required for instance, if a physical connection cannot be activated functionally. The allowed successors and predecessors of the instruments are determined by augmenting the implicit security specification with the explicit specification [KSRG$^+$17]. It considers the trustworthiness of the IP-cores and the information confidentiality and is provided as a list of instrument pairs. For each instrument pair in the list, it is stated, whether the connectivity between the instruments is explicitly allowed or explicitly restricted. If a certain connectivity between the instruments is explicitly restricted, the information about this connectivity is saved and is further used by the automated resynthesis. By contrast, if a connectivity between the instruments through the RSN is explicitly allowed although no functional path is sensitizable between these instruments in the initial design, an additional edge is introduced into the instruments graph.

Improper RSN integration might introduce additional unwanted connectivities between the instruments of the original system. These connectivities are referred to as security compliance violations and are defined as follows.

Definition 6.1 (Security Compliance Violation). A *Security Compliance Violation* $viol(p_j, p_k)$ is defined as a connectivity between the primitives p_j and p_k, which extends the allowed connectivity of the original system. The primitive p_j is called the source, and the primitive p_k is the destination of the violation.

For each pair of primitives (p_j, p_k), there exists at most one violation from v_j to v_k. It considers violating connectivities between these primitives through one or multiple paths. The absence of unwanted connectivities due to improper RSN integration is formulated in Formula 6.8:

$$\sum_{p_j \in P} \sum_{p_k \in P} viol(p_j, p_k) = 0 \tag{6.1}$$

6.1.2 Reachability Analysis of RSNs

An accurate functional reachability analysis using conventional methods is time-consuming due to complex control dependencies and possible interaction with the system logic. Also, the number of sensitizable scan configurations can be exponential in the number of control elements, such as scan multiplexers or SIBs.
The presented approach effectively overcomes the challenges above and computes the functional reachability of RSNs by performing the following steps:

1. Structural dependencies are determined.
2. The possible assignments of control signals are analyzed to identify a subset of dependencies which belong to a valid scan configuration.
3. The dependencies between valid scan configurations due to retargeting are determined and functional reachability of the RSN is computed.
4. The connectivities between the instruments through the RSN are determined.

Structural Reachability Analysis

Although functional cycles or cyclic active scan paths are considered as a bad practice by IEEE Std. 1687 [IEE14], structural cycles may occur. The existence of structural cycles in the RSN graph is checked by a Depth-First-Search algorithm. If the graph does not contain any cyclic dependencies, the original RSN graph is used as its acyclic representation.

Otherwise, an acyclic representation is constructed by removing a small number of edges from the RSN graph, in a similar way as it is well-known in partial scan design [KW90]. The information about all the cycles in the graph is preserved. All the vertexes of the acyclic representation are ordered topologically. The pairwise structural connectivities between the scan primitives are identified by a Breadth-First-Search routine. If the RSN graph contains cycles, the reachability is adjusted. For each edge, which has been removed from the original RSN graph to obtain an acyclic representation, the reachability of its source and destination vertexes is computed.

Control Signals Analysis

Direct data transfer in RSNs is only possible between the scan primitives, which are currently included into a path. The control signals are used to drive the explicit "*select*"-ports of the scan primitives and to determine the activated input of a scan multiplexer. The control dependencies are analyzed in an RSN graph to determine the connectivities which are functionally activated within a single scan configuration.

Definition 6.2 (Essential Select Condition)**.** The *Essential Select Condition (ESC)* $ESC(v_j, sig_1, ...sig_n)$ for a given vertex v_j is a Boolean formula in Conjunctive Normal Form (CNF), which defines the required group of assignments to the control signal values $(sig_1, ...sig_n)$, such that the vertex v_j is included into an activated path.

The *Essential Select Conditions (ESCs)* are iteratively computed starting from the sink vertex. The computation traverses the RSN graph backward to the root vertex in a Breadth-First-Search-manner. For each vertex, only those control signals are added into its ESC, which are required to place this vertex into an active scan path.

If a scan primitive, corresponding to the vertex, is directly connected to a multiplexer, the ESC demands a specific input of the multiplexer vertex to be selected in order to propagate the data from the given vertex to the scan-out vertex. This part of the ESC is defined by the *Relative Select Condition (RSC)*. The $RSC(v_j, v_k, sig_1, ...sig_n)$ for a given vertex v_j and a scan multiplexer vertex v_k, is a Boolean formula in CNF. If v_k is a direct successor of v_j, it defines a group of assignments to the control signals $(sig_1, ...sig_n)$, which are required to control the vertex v_k, in a way that the vertex v_j is included into the path.

The ESC of each vertex v_j depends on the ESCs and the RSCs of all of its n direct successors $(v_{l1}, ...v_{ln})$ and is computed using the following formula:

$$\begin{aligned} ESC(v_j, sig_1, ...sig_n) := \bigvee_{k=1}^{n} [&ESC(v_{lk}, sig_1, ...sig_n) \\ &\wedge RSC(v_j, v_{lk}, sig_1, ...sig_n)] \end{aligned} \tag{6.2}$$

According to this equation, a given vertex v_j is selected into an active scan path if and only if one of its direct successors is selected $(ESC(v_{lk}, sig_1, ...sig_n))$. If the selected successor models a scan multiplexer, then its scan-in branch, which includes the vertex v_j must be selected by the control signals $(RSC(v_j, v_{lk}, sig_1, ...sig_n))$.

Valid Scan Dependencies Analysis

Valid scan dependencies analysis is formulated as an instance of a so-called Boolean satisfiability problem, as defined below.

Definition 6.3 (Boolean satisfiability). A *problem of Boolean satisfiability* (SAT) is formulated as follows: Given a Boolean formula $f(x_1, \dots x_n)$; $x_i \in \mathbb{B}$ it is to determine whether there exist at least one assignment to variables $(x_1, \dots x_n)$, which evaluates the Boolean formula to the value of logic one.

If such an assignment exists, a formula is referred to as a *satisfiable* Boolean formula. Otherwise, the formula is *unsatisfiable*. For solving a SAT-problem, a Boolean formula is often represented in a Conjunctive Normal Form (CNF). The first complete solution to solve the Boolean satisfiability problem has been proposed by the Davis-Putnam (DP) algorithm [DLL62]. Thanks to an extensive research and development of SAT-based algorithms [BFH^{+}21], performance of the existing heuristic SAT-algorithms is rather high, and mapping a problem instance to a instance of Boolean satisfiability (SAT) problem provides quite an efficient way to solve many decision problems in the field of digital design and optimization. At the same time, one should not forget that the SAT-problem was the first problem, which has been proven to be NP-complete.

Therefore, additional efforts are required to formulate a SAT instance for a given problem in a way that the investigated solution space of the algorithm is reasonably small. For the problem of valid scan dependencies analysis, the SAT instance is formulated as follows.

The data transfer within one CSU-operation from a source vertex v_j to a destination vertex v_k is possible, only if at least one valid assignment to control signal values exists. In this assignment both vertices are selected in a valid active scan path, and the vertex v_k is reachable from the vertex v_j. If the vertices v_j and v_k fulfill the condition above, v_j is called an ASP predecessor of v_k, and v_k is called an ASP successor of the primitive v_j.

To verify the condition above, the *Essential Select Conditions* for the structurally connected vertices v_j and v_k are combined by conjunction. The existence of an assignment to the control signals $(sig_1, ...sig_n)$, which satisfies the boolean satisfiability (SAT) instance below, is verified:

$$\begin{aligned} \exists(sig_1, ...sig_n) : [&ESC(v_j, sig_1, ...sig_n) \\ &\wedge ESC(v_k, sig_1, ...sig_n) \\ &\wedge (v_j \in sp(v_k, G^{RSN})] \end{aligned} \tag{6.3}$$

- If the SAT instance is satisfiable, then an active scan path including both vertices can be configured. The satisfying assignments provide the essential values of logic signals to select both vertices simultaneously. The sets of *ASP predecessors* for v_k and *ASP successors* for v_j are correspondingly updated, since the data can be transmitted between v_j and v_k within one CSU-operation.
- If the SAT instance is unsatisfiable, then the ESCs of the vertices are contradicting and an active scan path traversing both vertices does not exist.

Functional Reachability Analysis

Retargeting in RSNs is used to propagate data between the vertices using multiple sequentially activated paths. The connectivities within individual activated paths, which have been computed as shown above, are generalized to determine the functional successors and predecessors of each vertex of an RSN graph. The number of reconfigurations used to propagate the data between two vertices and thereby the computation efforts in the worst case are limited to the sequential depth of the analyzed RSN, which is defined as the length of the longest possible active scan path inside the RSN.

6.1.3 Hybrid Connectivities Computation

To compute the connectivity properties after the RSN integration, the hybrid paths traversing both the instruments graph and the RSN graph are identified. Therefore, the transitive closure over the system graph is computed by combining the previously determined connectivities. For simplicity, the vertices of the instruments graph are referred to as the instrument vertices. The vertices in the RSN graph are referred to as the RSN vertices. The presented algorithm to compute the dependencies between the instrument vertices and the RSN vertices has rather low complexity, and contains two basic steps detailed below:

- **Bridge-dependencies**: The connectivities between the RSN and the instruments subgraphs are augmented with the connectivities within the subgraphs to build the hybrid partial paths between the instruments and the scan segments. These connectivities are referred to as bridges. Following the transitivity property, all the vertices in the RSN graph, which are reachable from the RSN vertex reading the data from the instrument vertex, are also reachable from the instrument vertex itself. In this way, all the vertices in the RSN graph are identified, which are functionally reachable from each given instrument vertex through the RSN graph. Following the same logic, any instrument vertex reading the data from a particular RSN vertex is reachable from all the vertices in the RSN graph, which reach this vertex.

  Example: *In Fig. 6.1, the scan segment s_1 is reachable from the instrument i_1. This means that all the scan segments, which are functionally reachable from s_1, are also reachable from i_1 through the RSN.*

- **Instrument connectivities through the RSN**: The partial paths from the first step are combined to find the connectivities between the instrument vertices through the RSN graph. Connectivity between the source and the destination instrument vertices exists through the RSN graph, if at least one intermediate RSN vertex exists,

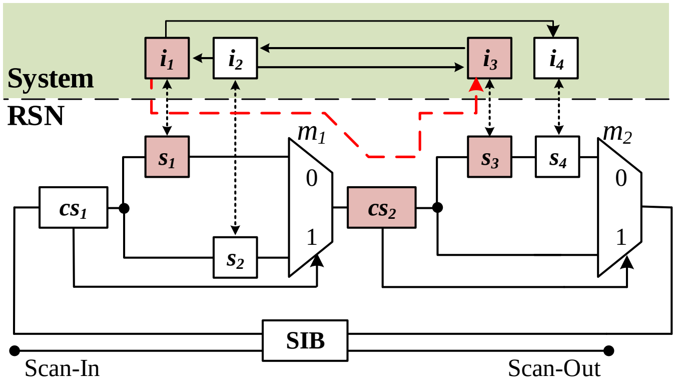


Figure 6.1: Security compliance violations in the RSN example

which is reachable from the source instrument vertex, and such that the destination instrument vertex is reachable from this RSN vertex.

Example: *In Fig. 6.1, the scan segment s_3 is reachable from the instrument i_1, and i_3 is reachable from s_3. This implies that i_3 is reachable from i_3 through the RSN.*

The identified connectivities between the instrument vertices through the RSN graph are augmented with the allowed functional connectivities within the instrument graph. It allows obtaining information about hybrid paths which can traverse both parts of the system graph an unlimited number of times. As a result, for all the instrument vertices, the sets of functional successors $fs(v_j, G^S)$ and predecessors $fp(v_j, G^S)$ after the RSN integration are identified.

6.1.4 Identification of Violating Connectivities

The compliance of the RSN with the initial security properties of the system is verified. The intended sets of allowed successors of the instrument vertices in the instruments graph are compared with the actual sets of successors in the combined system graph. The hybrid connectivities after the RSN integration must not exceed the allowed connectivities in the initial system:

- If the requirement is fulfilled for all the instrument vertices, the initial RSN is compliant with the initial security properties and the integration of the RSN is complete.

- If for any instrument vertex this requirement does not hold, the initial RSN is structurally modified to ensure its security compliant integration into the system. At this step, the list of security violation warnings is constructed and provided to the automated resynthesis.

Example: *In Fig. 6.1, as a result of the security compliance verification, the violating connectivities from the vertex s_1 to s_3; from s_2 to s_4 and from s_3 to s_4 are identified.*

6.2 Synthesis of Security Compliant RSNs

This section presents two approaches to resolve the security compliance violations, which might be introduced into the original system due to improper RSN integration. The remainder of the section is organized as follows:

- In Section 6.2.1, a purely structural method is presented to cut all the violating connectivities without sacrifying the allowed accessibility of the instruments via RSNs.
- In Section 6.2.2, this method is further enhanced to allow security compliant access for multiple user groups. The developed method preserves all the benefits of the purely structural solution and also dramatically reduces the hardware costs compared to the initial solution.

6.2.1 Structural Security-Compliance Aware Resynthesis

In this section, security compliance violations due an improper RSN integration are resolved by applying the structural resynthesis, which has first been published in [LAWW20] and enhanced in [LWW22d]. The general flow of the security compliance analysis and resynthesis (SCAR) approach is shown in Fig. 6.2 and consists of the following steps:

- **Analyze the security compliance**: The security compliance of a given RSN with the security properties of the initial system is analyzed as shown in Chapter 6.1. The connectivities in the RSN, which extend the allowed connectivity between the instruments of the initial system, are identified as security compliance violations.

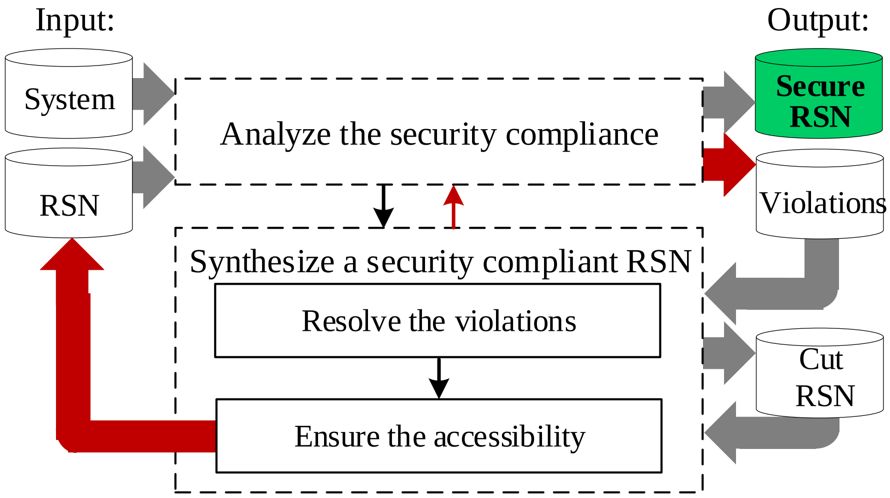


Figure 6.2: General flow of SCAR

- **Resolve the violations**: The identified violations in the RSN are resolved by using an efficient heuristic. A minimized set of structural changes is identified and applied to the RSN structure in order to eliminate all the violations and to prevent information leakage.
- **Ensure the accessibility**: If certain scan segments are not any more accessible after the previous step, some additional connectivities are added, and a modified RSN is constructed.
- **Validate the security compliance** (shown with a red arrow on the left): The security compliance analysis is performed once again for the modified RSN. If any violations are present in the RSN, the procedure above is repeated until all the violations are resolved. SCAR is guaranteed to converge with a security compliant RSN, since in the worst case a parallel RSN structure is obtained, where each instrument is accessed via a separate branch of a scan multiplexer. In practice, much less changes are required.

Eliminating the Violating Connectivities

The problem of eliminating the violating connectivities in RSNs is solved by using Integer Linear Programming (ILP). This subsection first formally defines ILP. Next, the problem formulation is presented and the steps to obtain a solution to the problem are provided.

Integer Linear Programming

In linear optimization, all the requirements, including the objective function and the constraints, are represented by linear relationships. If all the unknown variables x are additionally required to be integers, a considered problem instance is referred to an Integer Linear Programming (ILP) instance [Sch86]. In general, an ILP problem instance can be formulated as follows:

$$\begin{aligned}
&\underset{x\in X}{\operatorname{argmin}}[\sum_{j=1}^{n} c_j x_j] \\
&\sum_{j=1}^{n} a_{i,j} x_j \leq b_i \\
&j = 1, 2, \dots n, i = 1, 2, \dots m \\
&x_j \geq 0, x_j \in \mathbb{Z}
\end{aligned} \tag{6.4}$$

where x_i is an Integer variable of an n-variable vector x in range X; c_j is a weight coefficient, $a_{i,j}$ is a weight coefficient and the formula above is an inequality constraint.

Additionally, equality constraints can be added. For many problems in the field of digital design and resynthesis, variables are required to be 0 or 1 rather than arbitrary integers. In this case, a problem instance is referred to as a *0-1* or *Binary Linear Programming*.

Modern ILP solvers, such as Gurobi [Gur19], support efficient solving of ILP problem instances. They exploit, e.g. branch-and-bound or cutting plane methods. However, finding an exact solution for an ILP problem instance in the worst-case requires exponential runtime (NP-hard). If some domain-specific properties of the solution space are known, they can be formulated as additional constraints in order to reduce the investigated solution space but still keep the solution accurate. Additionally, a number of heuristic approaches, such as divide-and-conquer and dynamic programming, can be applied to significantly speed-up the solving process and still obtain an acceptable solution.

Minimum Cut Problem in a Multi-Commodity Flow

Resolving security violations can be mapped to a cutting problem in a directed graph. A single commodity $(vs - vt)$ in a directed graph $G := (V, E)$ is a vertex pair, where vs is a source and vt is a destination. A subset of vertices V_{cut} is called a $(vs - vt)$ cut, if its removal from the vertex set V would remove the connectivity from the source vs to the destination vt in the resulting graph.

If k commodities $(vs_k - vt_k)$ coexist in a graph G, a cut in a multicommodity flow graph can be defined as a vertex subset V_{cut}, such that for all the commodities $(vs_k - vt_k)$ the connectivity would be precluded [FF58, HHS07].

Problem Formulation

The problem of automated RSN resynthesis is formulated as a minimum cut problem in a multicommodity flow and is solved by means of Integer Linear Programming. Since this problem is NP-complete [HHS07], an efficient and precise divide-and-conquer heuristic is applied.

The presented algorithm includes the following steps:

- The list of security violation warnings is processed, and each violation $viol_x(v_j, v_k)$ is mapped to a single commodity. Mapping to a single commodity is possible, since each violation represents the existence of connectivity between the corresponding vertices, possibly through multiple paths.

- An initial vertex cut $V_{cut}^{RSN} \subset V^{RSN}$, which would remove the connectivities for all the violations is computed. For each intermediate vertex v_m of the RSN graph, which belongs to at least one violating functional path between the source v_j and the destination v_k, we decide whether (Fig. 6.3):

    1. v_m belongs to the cut itself, or
    2. all the paths between v_j and v_m have to be cut, or
    3. all the paths between v_m and v_k have to be cut.

    In Fig. 6.3, a single intermediate vertex v_m is removed to cut all the paths between v_j to v_k.

- The solution is adjusted recursively to identify a possibly small vertex cut. The experimental results show that the presented heuristic resolves a large number of violations by applying just a few structural changes and is scalable for large RSNs.

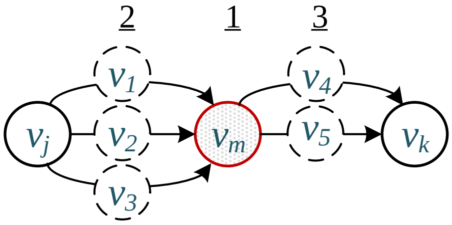


Figure 6.3: Node cutting options

- From the connectivity perspective, the removal of a vertex is equivalent to the removal of all its outgoing edges. To preserve all the vertices, which correspond to the scan segments, all the outgoing edges of the vertices in the cut are removed from the graph instead of removing the vertices themselves.

- The accessibility of the scan segments, and thereby of the corresponding instruments, can be affected after removing the violating connectivities in the RSN graph. The accessibility of such a scan segment is ensured by the method described below.

Base Step

Each violation $viol(v_j, v_k)$ is modeled as an edge between the source vertex v_j and the destination vertex v_k. The intermediate vertices between the source and the destination are not considered at this step. Such RSN representation is referred as the *Level-0 graph*, as shown in Fig. 6.4.a for the RSN from Fig. 6.1, where the set VS contains all the violations' sources, and the set VT contains all the destinations. These sets are not forced to be disjoint, and a source of one violation can serve as a destination of another violation. To resolve the violations, the paths between the sources and the destinations are cut, and the following steps determine where.

Recursive Step At each further step with an index n, an optimized intermediate set $OINT_n$ is constructed. It contains a small number of vertices, such that for violating path, there exist at least one vertex v_l whose removal would cut this path.

We further say that v_l covers this path. The intermediate vertex set $OINT_n$ is placed in the graph between the sources VS and the destinations VT of the considered violations. This more accurate RSN representation is referred as the *Level-n graph* G_n^{RSN}, as shown in Fig. 6.4.b for the RSN from Fig. 2.1, and is used to adjust the accuracy of the solution.

To compute the set $OINT_n$, first the optimized sets of the intermediate vertices $OINT_{n,x}$ are built for each specific violation $viol_x(v_j, v_k)$. Each set includes a small number of vertices, whose removal would cut all the functional paths, which cause this violation. An empty set of covered vertices $COV_{n,x}$ is initialized to keep track on the covered paths.
For each Level-n graph, an unoptimized intermediate set INT_n includes all the vertices belonging to the violating paths. Since the size of the optimized set only affects the runtime performance of the resynthesis, an efficient heuristic is proposed to reduce its size:

- Firstly, a global weight $weight_G(v_l, G_n^{RSN})$ is defined for each vertex $v_l \in INT_n$ in the Level-n graph. It shows, how often is the given vertex v_l reachable from any of the violation sources VS, and thereby considers all the violations simultaneously.

- Secondly, for each vertex $v_l \in INT_n$ a local weight $weight_L(v_l, viol_x(v_j, v_k))$ is calculated with respect to the violation $viol_x(v_j, v_k)$. All such vertices v_m are identified, which belong to at least one sensitizable functional path between the source v_j and the destination v_k. Out of these vertices, such vertices are selected that a connectivity between v_l and v_m exists. Their quantity is normalized with respect to the total number of vertices, which belong to any path between v_j and v_k. The resulting local weight defines how many functional paths between v_j and v_k are covered with the given vertex.

- Thirdly, a "memory"-function $mem(v_l, G_n^{RSN})$ defines how often a certain vertex has been already included into the optimized set of intermediate vertices for the previously processed violations.

- Lastly, a cost function, which depends both on the global and the local weight as well on the value of the memory function, is computed:

$$\begin{aligned} cost_(v_l, G_n^{RSN}) := \Phi[&weight_G(v_l, G_n^{RSN}), \\ &weight_L(v_l, viol_x(v_j, v_k)), \\ &mem(v_l, G_n^{RSN})] \end{aligned} \tag{6.5}$$

A vertex with the highest value of a cost function is selected and is added into an optimized set $OINT_{n,x}$ of a considered violation and into the set of covered vertices $COV_{n,x}$.

All the functional successors and predecessors of this vertex are also added into the set of covered vertices $COV_{n,x}$. The value of the "memory"-function is incremented for the selected vertex, while the global and local weights do not need to be recomputed. Depending on the new value of the cost function, the next vertex is selected and the sets of intermediate and covered vertices are updated. The procedure is repeated, until all the intermediate vertices for a violation are covered.

The same procedure is performed for all the violations, in a way that the decision for each further violation considers the previous decisions for the already processed violations.

The resulting optimized set of intermediate vertices $OINT_n$ is built as a union of the computed sets for single violations:

$$OINT_n := \bigcup_{x=1}^{\#violations} OINT_{n,x} \tag{6.6}$$

For each vertex v_l from the set $OINT_n$, which belongs to at least one violating path between the vertices v_j and v_k, it is decided whether the paths from v_j to v_l, or the paths from v_l to v_k are cut, or the vertex itself is removed.

The minimized set of connectivities, whose removal would resolve all the violations for the current graph representation G_n^{RSN}, is selected by solving a minimum cut problem in a multicommodity flow on a smaller graph by means of Integer Linear Programming (ILP).

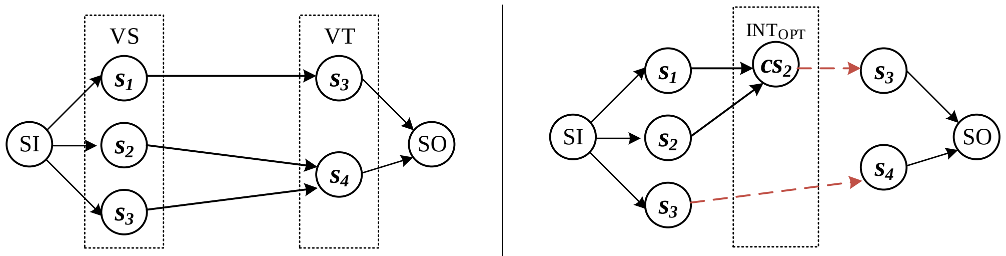
Final Steps and Termination

Assume that at the recursive step n, the functional paths $(path_1, ..path_m)$ are cut. Then, at the next step, it is decided, where exactly these paths must be cut. The solution accuracy is improved incrementally at each recursive step and the lengths of the violating paths are gradually decreasing. The number of recursive steps depends on the graph size, and on the maximum distance between the source and the destination of a violation.

At the recursive step $n + 1$, the first vertices of the paths, which have been cut in the previous step, are considered as the violations sources, whereas the last vertices in the paths - as the destinations of the violations. A minimized number of the intermediate vertices are added between the sources and the destinations. As a result, the graph G_{n+1}^{RSN} is constructed, and the minimum cut flow problem is solved again. The computation converges when the exact vertices of the high-granular graph G^{RSN} are in the cut V_{cut}.

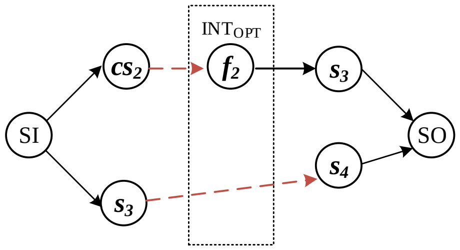
As mentioned before, instead of removing the vertices, all the outgoing edges of the vertices of the cut V_{cut} are removed from the RSN graph. The secure RSN graph is built, where some edges are removed to resolve all the violations.

Example: *Fig. 6.4 describes the required steps for the RSN in Fig 2.1 to cut all the violating connectivities.*



a) Level-0 graph

b) Level-1 graph



c) Level-2 graph

Figure 6.4: Graph construction

a) *Level-0 graph*: *The vertices s_1, s_2, and s_3 serve as sources of violations, while s_3, s_4 serve as destinations. Auxiliary vertices SI and SO serve as a global source and destination. s_3 serves as a source of one violation and as a destination of another violations.*

b) *Level-1 graph construction*: *The vertex cs_2 has the highest value of a cost function and covers all the violating paths from s_1 to s_3. After cs_2 is added into the optimized set, the value of $sel(cs_2)$ function is incremented. The same vertex cs_2 is automatically selected to cover the paths from s_2 to s_3. The remaining violation from the vertex s_3 to the vertex s_4 is resolved by removing the edge inbetween.*

c) *Level-2 graph is constructed: The paths removed in the previous step (b) are considered as violations. To cover the violation between the vertices cs_2 and s_3, the vertex f_2 is selected. The edge between the vertices cs_2 and f_2 is removed to resolve the remaining violations (s_1 to s_3 and s_2 to s_4). The computation converges since the solution cannot be further adjusted.*

As shown in Fig. 6.5 for the RSN from Fig. 2.1, the edges from s_3 to s_4, and from m_1 to cs_2 are removed from the graph to resolve the violations.

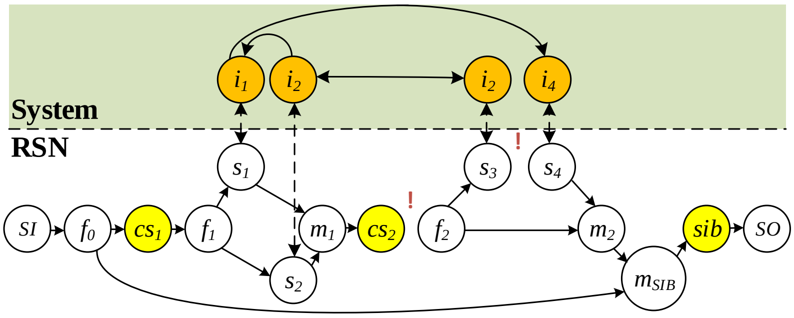


Figure 6.5: Secure RSN graph

Reintroducing the Accesibility

To eliminate the violating connectivities, some edges are removed from the RSN graph. Therefore, some scan segments as well as the corresponding instruments may become inaccessible.

The accessibility of the scan segments is re-installed in an automated way and a minimized number of novel connectivities is added sequentially into the RSN graph. The number of added edges is at most $2 * m$, where m is the number of previously removed edges. The connectivities, which are added into the RSN graph, must fulfill the following conditions:

- Each newly introduced connectivity is compliant with the security properties of the design-under-test.
- After augmenting the RSN graph with additional edges, the accessibility of all the scan segments, is guaranteed through at least one sensitizable active scan path.

The existence of a sensitizable path $path(SI, v)$ from a primary scan-input to a given vertex v is further referred to as *backward accessibility*, whereas the existence of a path $path(v, SO)$ from a vertex v to a primary scan-output is called *forward accessibility*. If the vertex is forward and backward accessible, and if the activation conditions of the subpaths $path(SI, v)$ and $path(v, SO)$ are not contradicting, then a path from a scan-in through a given vertex to a scan-out exists, and the vertex is accessible.

To ensure the accessibility, first, the *forward accessibility* of the vertices is ensured, as summarized in Algorithm 6.1. To verify the existence of a path from a given vertex to the primary SO, the vertices are traversed in a reversed Breadth-First-Search (BFS)-order, which starts from the primary scan-in port (*Lines 1-2*).

Algorithm 6.1: Ensuring the forward accessibility

```
1 $VertexOrder := Reverse(Order(V^{RSN}, BFS, SI))$;
2 for $v_j \in VertexOrder$ do
3     if $fs(v_j, G^{RSN}) = \emptyset$ then
4         Find allowed functional successors;
5         Select successor based on $optimizationcriteria$;
6         Update the reachability properties;
7     end
8 end
```

- *Line 4*: The candidate vertices v_k are identified, which can serve as possible successors of the vertex v_j. Adding a connectivity from the given vertex v_j to v_k must be compliant with the security properties of the design, and a functional path from v_k to the primary scan-out should exist. Here, not only a direct edge between v_j and v_k must be considered, but also the connectivities induced by any combinations of the functional predecessors of v_j and the successors of v_k in the combined system graph. The candidate successors must be also compliant with the explicit security specification. If the connectivity between the vertices v_j and v_k introduces an explicitly prohibited connectivity into the design, the vertex v_k is not selected. The candidates set is guaranteed to be non-empty since it always includes the scan-out port.

- *Line 5*: The actual successor v_k is selected out of the candidates. The choice depends on the optimization criteria, such as the access latency or the hardware overhead, which is specified by a DfT integrator.

- *Line 6*: An edge is added between v_j and the selected successor v_k. The reachability of v_j and v_k, as well as the reachability of all the functional predecessors of v_j and the functional successors of v_k is adjusted to reflect the novel connectivity.

The process (*Lines 4 - 6*) is repeated to ensure the forward accessibility of all the affected vertices, and thereby the accessibility of the corresponding instruments.

The same idea is applied to guarantee the backward accessibility. The accessiblity of the segments with respect to the valid assignments to the control lines is verified as in [BKW15b].

Example: *An example for an accessible RSN graph is shown in Fig. 6.6 for the RSN from Fig.2.1.The edges from* s_3 *to* m_2*, from* f_2 *to* m_2*, from* f_0 *and* m_1 *to* m_{SIB} *are added to ensure the accessibility while preserving the security compliance.*

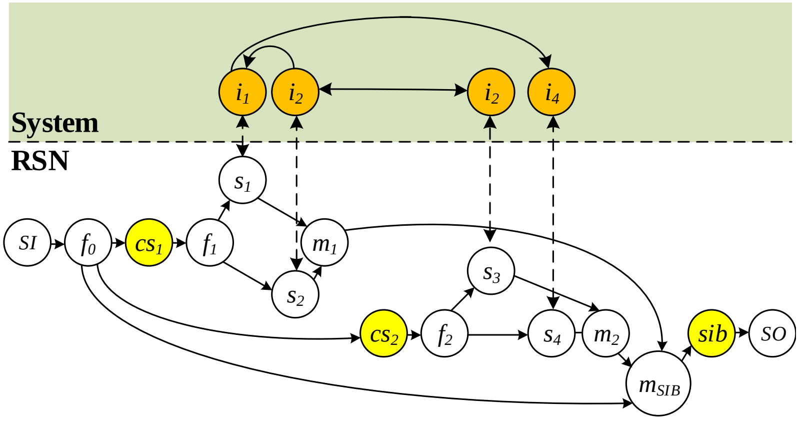


Figure 6.6: Accessible RSN graph

Compliance Validation

After applying the previously described heuristic, the existing violating connectivities are removed but some novel connectivities are added into the RSN to ensure accessibility. These additional connectivities may cause novel violations in the system. Iterative validation is used to guarantee that all the violations are resolved, as shown in Fig. 6.2 with a big red arrow on the left. After applying the resynthesis, the security properties of the resulting RSN are validated once again by using the security compliance analysis, described in Section 6.1:

- If the connectivities in the resulting RSN do not extend the allowed connectivity between the instruments of the original system, then the security-preserving RSN structure is already obtained, and this RSN is used to access the test instruments.
- If some violations are still present in the RSN, the heuristic is repeated again until a secure RSN implementation is synthesized.

The presented scheme guarantees to converge to a security-preserving RSN. In the worst case, each instrument is accessed via a scan segment, which is located on the individual branch of a scan multiplexer. In the experiments we show that the algorithm terminates much faster, and in most cases, a security compliant RSN is synthesized after the first iteration of SCAR.

6.2.2 Hybrid Protection Method for Multiple Users

Access or observation through specific functional or non-functional channels can be restricted. The security preserving integration scheme presented above ensures that the integrated RSN does not extend the allowed connectivity of the original system for the case, when the security specification is static and all users have unified access rights. However, to support Silicon Lifecycle Management, the access rights can be specified for various user groups, starting from the normal users, upto the manufacturer and the test engineers. E.g., some confidential data must only be accessible to the customer. Access to some diagnostic instruments must only be eligible for the test engineers.

The access specification for multiple users can be specified as follows. For a user u_j a subset of primitives $G(u_j) \subset P$ can be specified, such that u_j must not access $\forall p_k \in G(u_j)$. The subset $A(u_j) \subset P$ defines all primitives, which must remain accessible for u_j.

Security violations for multiple user groups are formulated as follows.

Definition 6.4 (User-specific security compliance violation)**.** A *user-specific security compliance violation* $viol_u(p_j, p_k)$ is a connectivity from the source p_j to the destination p_k, which extends the functional connectivity of the system or violates the explicit specification for at least one user.

Definition 6.5 (Authorization violation)**.** An *authorization violation* $viol_u(p)$ is an unfulfilled requirement to restrict an access to a $p \in G(u)$ for u.

If multiple users with different user rights are specified in an RSN, the integration scheme must consider it. The security compliance violations must be resolved for all the users.

$$\sum_{u \in Users} \sum_{p_j \in P} \sum_{p_k \in P} viol_u(p_j, p_k) = 0 \tag{6.7}$$

where $viol_u(p_j)$ is an unwanted extra-connectivity for user u.

It must be guaranteed that only eligible users access the instruments:

$$\sum_{u \in Users} \sum_{p \in P} viol_u(p) = 0 \tag{6.8}$$

where $viol_u(p)$ equals to 1 if a user u is not eligible to access a scan primitive p.

Although the resynthesis method presented above efficiently resolves security compliance violations for a single user, it cannot be used to apply user-specific restrictions for accessing the instruments. As the existing methods cannot be readily applied to solve this problem, as detailed below, an enhanced protection method, which has been first published in [LAW21] is presented in the current section.

Limitations of the Existing Protection Methods for Multiple User Groups

This section discusses why the existing schemes, including the one presented in the first part of the chapter, cannot be used to ensure protection for multiple users with diversified access rights. It also explains how the hybrid protection scheme presented below combines the benefits of the existing methods and provides a remedy over their limitations.

Filter-based protection

The sequence filter, as presented in [AKS+18], is a flexible way to resolve the violations online. It is constructed based on the RSN structural description and its specification, and is put between the TAP controller and the RSN, as shown in Fig. 6.7.

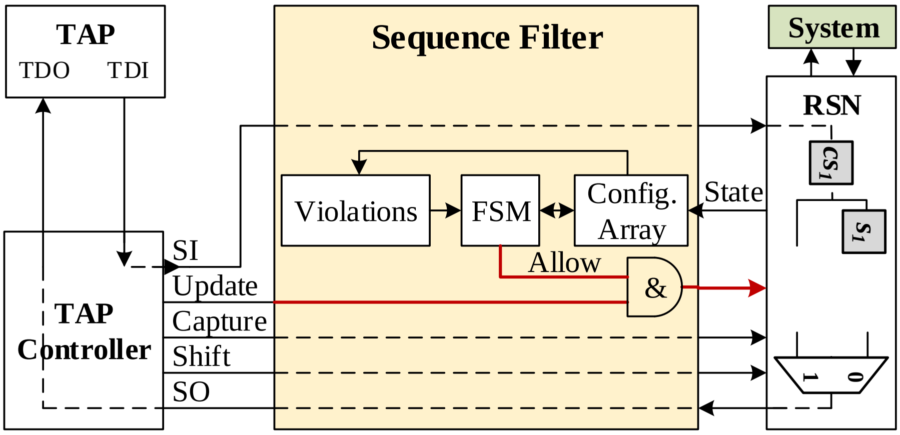


Figure 6.7: Details of the filter-based protection

The *Violations* are obtained as a list from the security compliance analysis, considering the explicit and implicit specifications, and stored as conditions into a logic block "Violations". A *Configuration Array* stores the RSN configuration. A *Finite State Machine* (FSM) captures the configuration bits from the input stream at the SI port and allows to keep the configuration array for the current specification updated. It allows to consider complex access requirements for multiple user groups. The filter-based protection does not require any modifications of the RSN structure, even if the security specification changes, e.g. if the trustworthiness of a third-party IP or the information confidentiality have changed.

Filters of access patterns can allow just a static set of precomputed accesses, as in [BKW14], or provide access protection for complex access scenarios, as in [AKS$^+$18]. Benefits of the filter-based approach are:

- Minimal hardware overhead nearly independent of the RSN complexity.
- Compliant with extensions of the RSN standards like the P1687.1 proposal [CVTR20] which defines access to RSNs through alternate interfaces.
- It is flexible and can be adopted for changing security requirements.
- Together with a standard authorization scheme, it can handle different access rights for different user groups.

A severe drawback of the filter approach is the fact that there may exist security violations which cannot be resolved without unwanted side-effects. Fig. 6.8 shows an example, where any filter approach would sacrifice the accessibility of other instruments as well.

Example: *Consider the RSN example from Fig. 2.1. Its graph is given in Fig. 2.7. The violating connectivities from the vertex s_1 to s_3; from s_2 to s_4 and from s_3 to s_4 are identified by the security compliance analysis algorithm, as shown in Chapter 6.1. An access specification is formulated as follows:*

- *An access to a safety-critical instrument i_3 is required for all user groups.*
- *An access to i_4 is allowed for test engineers, but is restricted for regular users.*

The existing methods fail to integrate an RSN in a security-preserving way, such that the access specification above is satisfied. First, the filter-based protection is considered.

If access to the safety-critical instrument i_3 *is required for all user groups, but the instrument* i_4 *should not be accessible for regular users, using the filter, as shown in Fig. 6.8 is not an option: it makes* s_3*,* s_4 *and thereby* i_3*,* i_4 *inaccessible through the RSN. All the paths traversing* s_3 *also traverse* s_4*, and restricting an access to one of them implies a restriction for the second one.*

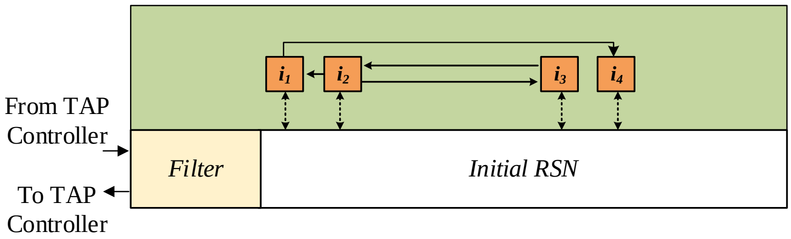


Figure 6.8: Limitations of filter-based schemes for the RSN from Fig. 2.1. Limiting the accessability to the instrument i_4 restricts the access to the instrument i_3.

Resynthesis of the RSN Structure

Structural resynthesis, as presented in Section 6.2.1, can resolve all security violations, but it comes with certain drawbacks as well:

- In some cases, relatively high hardware costs are incurred even when applying sophisticated synthesis procedures as in [LAWW20].
- If security requirements are changing, a complete resynthesis is necessary.
- User group specific access rights cannot be given.

Example: *Consider the same example, as discussed for the filter-based protection, but now for the structural resynthesis method. Resynthesis, as shown in Fig. 6.9, resolves this violation, preserving the accessibility, but does not consider various access permissions. If an access to* i_4 *is restricted for regular users, but not for test engineers, the RSN must be resynthesized again, implying even more HW costs.*

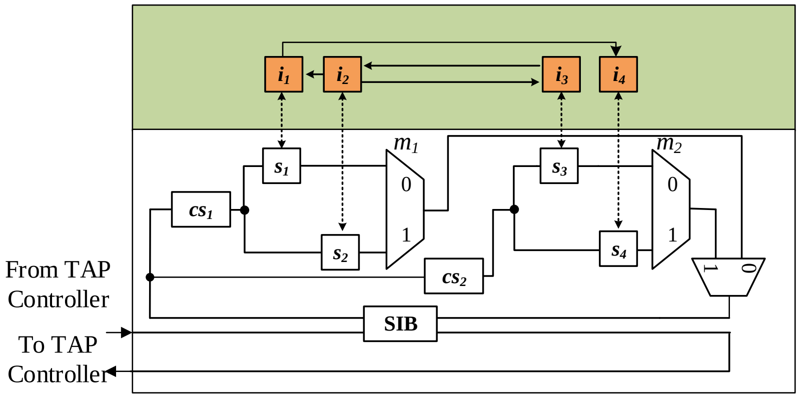


Figure 6.9: Limitations of resynthesis for the RSN from Fig. 2.1. Blocking the accessibility of the instrument i_4 only for certain user groups is not possible.

Hybrid Protection Scheme as a Remedy

While structural resynthesis can resolve all conflicts, it lacks flexibility. On the other hand, filters may sacrifice the required accessibility. The approach presented below avoids the drawbacks, its main contributions are:

- **Efficient analysis**: A filter applicability analysis is presented to identify a minimized subset of violations, which cannot be handled using filters, while preserving the accessibility of the instruments through the RSN.
- **Minimized hardware overhead**: The structural changes are applied to resolve only this small subset and to ensure that any further violation is resolvable using a filter.
- **Flexible access**: Most of the violations are resolved online by using a flexible filter, to ensure access to RSNs for various user groups with different access permissions.
- **Preserved accessibility**: All the required instruments remain accessible through the RSN for all user groups.

Example: *Let the same example and the same access specification be handled by the hybrid protection scheme, which is presented in the current section. A Hybrid Scheme, as shown in Fig. 6.10, considers various access permissions and the required accessibility is preserved. The RSN structure is slightly modified compared to the original RSN from Fig. 2.1, in order to make* s_3 *and* s_4 *accessible individually and thereby to ensure the instruments' accessibility. The remaining violations are resolved by the filter.*

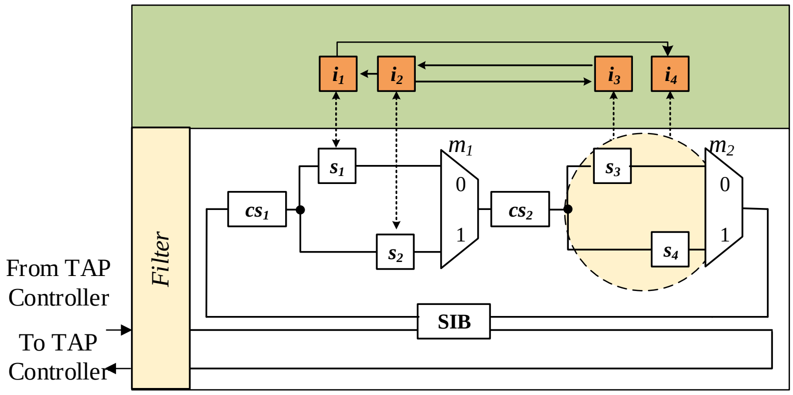


Figure 6.10: Hybrid Protection scheme as a remedy. For the RSN from Fig. 2.1 the access specification is fulfilled for multiple user groups.

Filter Compliance of an RSN

The majority of violations can be resolved by using a sequence filter but some scan segments may become inaccessible through the RSN despite a requirement in the specification, as already shown in Fig. 6.8. This section presents the first Filter Applicability Analysis, which identifies a maximized set of violations to be resolved by using a sequence filter without affecting the accessibility of other segments. For each vertex v_j, an Essential Select Condition $ESC(v_j, sig_1, ..sig_n)$ defines the assignment to the control signal values $(sig_1, ..sig_n)$, required to put this vertex into an activated path, and is represented in a conjunctive normal form (CNF). The essential condition is computed iteratively, starting from the scan-out *SO*, and considers the control signals required to select the appropriate branches of multiplexers and to trigger the *select*-signals.

Security Compliance Violation

A security compliance violation $viol_u(v_j, v_k)$ is called *resolvable* by a filter, if for each of the vertices v_j, v_k there exists at least one configurable path, which includes this vertex. The applicability of a filter to resolve such a violation can be checked by the following flow:

1. Compute the essential conditions $ESC(v_j, sig_1, ..sig_n)$, $ESC(v_k, sig_1, ..sig_n)$ and ensure that each of the vertices v_j, v_k is accessible through at least one active path.

2. Verify if data from v_j can be transmitted to v_k using a single scan configuration. A SAT instance is formed to find at least one assignment for the control signal values $sig_1, ...sig_n$, which put both vertices into a path simultaneously:

$$\begin{aligned}\exists sig_1,..,sig_n : ESC(v_j, sig_1, ..sig_n)\&ESC(v_k, sig_1, ..sig_n)\\ := True\end{aligned} \tag{6.9}$$

3. Verify if two paths can be configured, such that the first one is used to access v_j only, but should not traverse v_k.

   $$\begin{aligned}\exists sig_1,..,sig_n : ESC(v_j, sig_1, ..sig_n)\&\overline{ESC(v_k, sig_1, ..sig_n)}\\ := True\end{aligned} \tag{6.10}$$

   The second activated path is used to access v_k only.

4. If both paths can be configured, the violation is resolvable by a filter, otherwise it must be resolved structurally.

If it is not possible to assign control values such that Eq.6.9 is satisfied but v_j and v_k are functionally connected, the data transfer between these vertices requires retargeting by using more than one scan configuration. Hence, there are two paths, such that the first one traverses only v_j but not v_k, and the second path traverses v_k. This means that the violation is resolvable by a filter and a step 3 can be skipped. The same idea is used to verify, if the groups of the vertices $G_j(u_i), G_k(u_i)$ are accessible individually.

Authorization Violation

An authorization violation $viol_u(v_k)$ is called *resolvable* by a filter, if after applying a filter, for all the users u_l and for all the required vertices $v_{jm} \in A(u_l)$, there exist at least one activated path, which does not traverse the restricted vertex $v_k \in G(u_l)$, but traverses the required vertices $v_{jm} \in A(u_l)$. The applicability of a filter for a violation $viol_u(v_k)$ is verified using the same logic with v_k as a source and all sv_{jm} as destinations.

For each vertex $\forall v_{jm} \in A(u_l)$, which must remain accessible, an assignment for the control signals $sig_{1m}, ...sig_{nm}$ is searched, which includes v_{jm} into a path but does not include any of the restricted vertices $v_k \in G(u_i)$. If all such assignments are found, the authorization specification can be fulfilled by a filter without causing any authorization violation.

Example:

$$\begin{aligned} \exists sig_{1m}, ..sig_{nm} :& f(v_{jn}, sig_{1m}, ..sig_{nm}) \& \\ & [\cap_{k=1}^{|G(u_i)|} (\overline{ESC(v_k, sig_{1m}, ..sig_{nm})})] := True \end{aligned} \tag{6.11}$$

Access restriction to cs_1 for u_l (in Fig. 2.1), makes all the scan segments inaccessible through the RSN. If access to s_1 is required for u_l, the violation is unresolvable by using a filter.

Filter Compliance of an RSN

Definition 6.6 (Filter-compliant RSN)**.** An RSN is called *filter-compliant*, if all of the security compliance violations as well as all of the authorization violations are resolvable using a sequence filter without blocking the access to other segments.

The Filter Applicability Analysis is applied to all the violations sequentially in order to verify, whether a given RSN is filter-compliant. If the RSN is filter-compliant, the information about the violations can be further used to guide the filter generation. In the other case, the violating connectivities, which are unresolvable using a filter, can be resolved structurally by using resynthesis.

6.2.3 General Protection Flow

The hybrid protection scheme (Fig. 6.11) consists of two steps:

1. *Prepare the RSN*: an initial RSN is transformed into a *filter-compliant* RSN (FC-RSN in Fig. 6.11) using a minimized number of changes.
2. *Generate the Filter*: a sequence filter is generated for the filter-compliant RSN according to [AKS^{+}18] to resolve the remaining violations.

A secure RSN implementation, which preserves the specified accessibility of the segments, consists of a filter-compliant RSN and the generated sequence filter. This implementation combines the flexibility and online usability of the filter with the generality of the resynthesis, and allows to resolve any security compliance violation. The remainder of the section provides details on the individual steps.

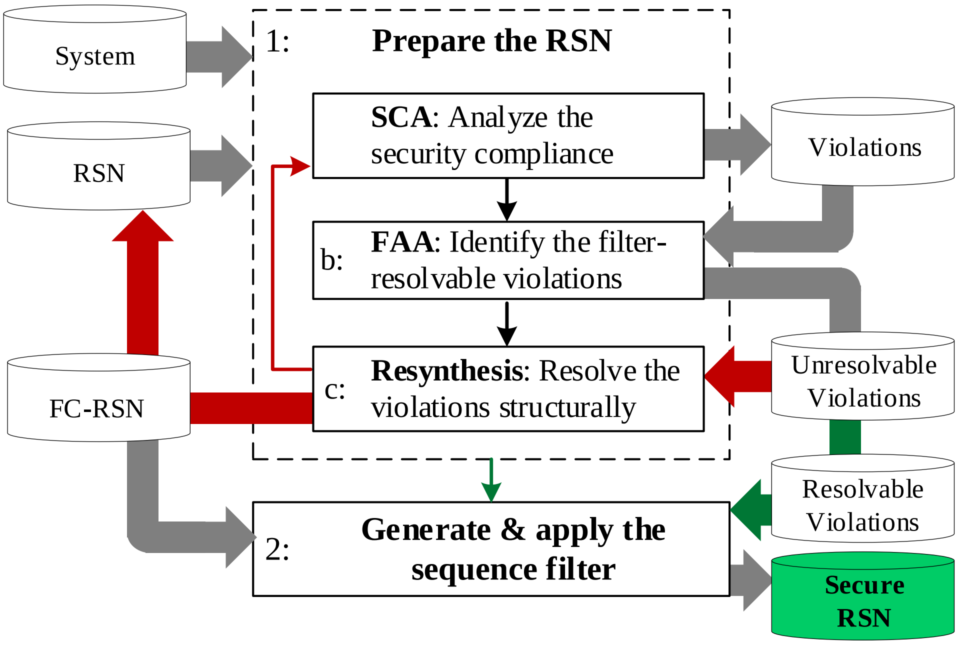

System
RSN
FC-RSN
1:
Prepare the RSN
SCA: Analyze the security compliance
b:
FAA: Identify the filter-resolvable violations
c:
Resynthesis: Resolve the violations structurally
Violations
Unresolvable Violations
Resolvable Violations
2:
Generate & apply the sequence filter
Secure RSN

Figure 6.11: General flow of the hybrid protection scheme

Prepare the RSN

Following steps are required to resynthesize the RSN:

1. **Security Compliance Analysis (SCA in Fig. 6.11):** The reachability properties of the initial system and the RSN are computed and the security compliance of the RSN with the system is verified. If the RSN is not compliant, the list of authorization and security compliance violations is generated.

2. **Filter Applicability Analysis (FAA in Fig. 6.11):** The list of the violations is analyzed using the Filter Applicability Analysis presented in Section 6.2.2, and the violations are divided into two subsets. The first subset includes the violations, which can only be resolved by applying the structural changes, the second subset - the violations resolvable by using a sequence filter.

3. **Resynthesis:** The resynthesis approach, as presented in Section 6.2.1, is adjusted to obtain a filter-compliant RSN with a minimized number of changes and is only applied to the subset of violations, which are unresolvable by a filter.

- A minimized number of edges is removed from G^{RSN} to cut the violating connectivities, which cannot be resolved using filters.
- The accessibility of the affected vertices is reintroduced sequentially.
- For each vertex v_j, which does not have successors, a prioritized list of new possible successors $PS(v_j)$ is formed. $PS(v_j)$ contains all such vertices v_k that a newly introduced connectivity between v_j and v_k would be resolvable by a filter, even if the specification changes.
- The highest priority is given to such resolvable connectivities, which do not cause a security compliance violation in the current security specification.
- If multiple vertices in $PS(v_j)$ have the highest priority, additional optimization criteria can be specified by a test engineer, e.g. a minimized access latency of safety-critical instruments or hardware overhead. For any vertex v_j, $PS(v_j)$ is not empty, since it includes the auxiliary Scan-Out vertex.
- The same idea is applied to reintroduce the accessibility of the vertices with no predecessors.

Validate the Filter Applicability

The RSN generated in step 1 is analyzed again to check, whether all the violations are already resolved and to generate the updated list of violations otherwise. The updated list is analyzed for the filter applicability and the subsets of violations are recomputed, since the structural changes can affect the filter applicability. The RSN is modified, until all the violations from the list are resolvable by the filter. This incurs minor number of structural changes, and offers flexibility for the future, if the specification will change.

The presented scheme is guaranteed to converge to a filter-compliant RSN and the remaining violations are resolved using a filter. In the worst case, a parallel RSN structure, where all the instruments are accessed using the scan segments, located in the different branches of a scan multiplexer, is obtained.

Generate and Apply the Sequence Filter

The list of violations, which have not been structurally resolved, and the modified filter-compliant RSN are used for the filter [AKS+18] generation. The filter handles the remaining violations and prohibits any violating access. The sequence filter together with a corresponding filter-compliant RSN, fulfill the security specification without affecting the accessibility of the required segments for multiple user groups with specific access rights. Secure user authentication can be accomplished, e.g. as in [PRML20]. The P1687.1 proposal [CVTR20] defines access to RSNs through alternate interfaces. The actual input signals are transformed using an FSM to keep compliance with the IEEE Std. 1687. In line with P1687.1, a TAP, augmented with a filter, represents a novel secure access interface.

6.3 Evaluation

This section evaluates the results of the developed approaches to ensure security-preserving RSN integration. The main goal of this section is to illustrate the efficiency of the developed concepts with the experimental data. Therefore, the diagrams in this section are presented for a limited but representative subset of benchmarks and mostly use relative normalized numbers. The detailed experimental data for all benchmarks, including the absolute numbers is provided in Appendix B.3. Section 6.3.1 presents the results for the developed security preserving integration scheme for a single user. In Section 6.3.2 the results for a hybrid protection scheme are provided.

6.3.1 Structural Resynthesis for Single User Protection

Each evaluated system models the connectivities between the instruments through the system, the connectivities inside an RSN, and the ones crossing the boundary between the RSN segments and the accessed instruments. The whole graph-based SCAR scheme is performed in a divide-and-conquer manner. The security compliance analysis is first run on the smaller sized blocks. Next, the reachability properties of these smaller blocks are merged to analyze the blocks with a larger size, until the whole RSN is processed. The identified violations are sorted: for each violation the smallest possible logical block is identified, such that both the source and the destination of the violation, as well as all the paths between them, are located inside this block. This information is used further by the automated resynthesis: if a certain violation is located inside an isolated block there is no need to resynthesize the whole graph. The violations are resolved hierarchically, starting from the smallest blocks. The violations between the blocks are handled at a higher granularity, such that, if possible, only the interconnects between the affected blocks are resynthesized to cut the violating paths.

Runtime Evaluation

The scalable divide-and-conquer processing made it possible to analyze large and complex RSN designs within an acceptable time. Fig. 6.12 shows the ratio between the runtime of the presented graph-based scheme and the previously published matrix-based approach [LAR$^+$19] for the RSNs from the ITC'2016 benchmark set. As the benchmark size increases, the runtime ratio increases up to 3.5 times. As expected, the memory consumption for processing the graphs is lower compared to the sparse matrix processing.

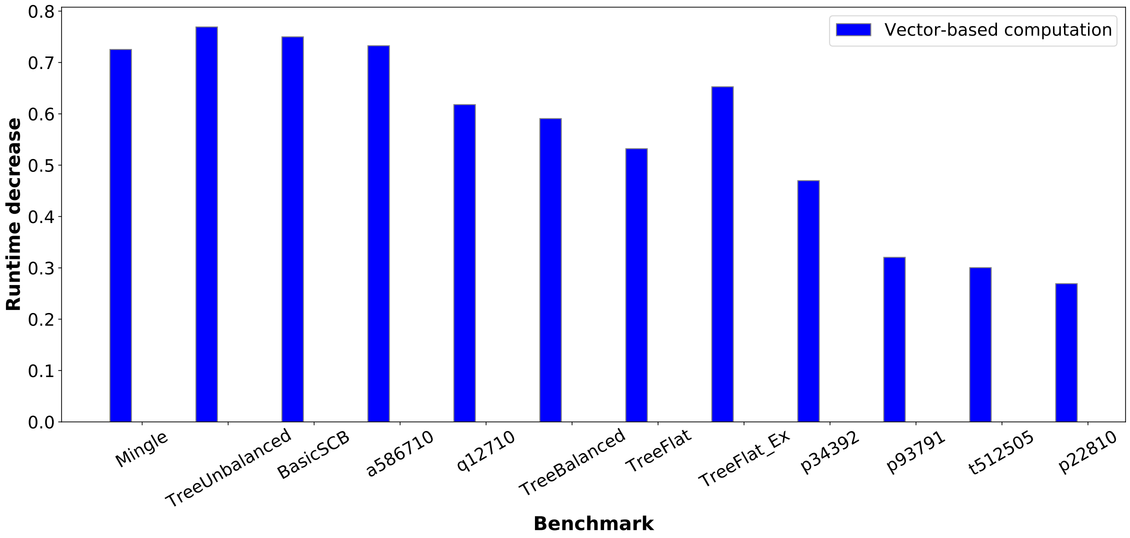


Figure 6.12: Runtime decrease compared to the matrix-based computation

For the DATE'19 benchmark set, the runtime improvement ratio for the presented approach compared to the matrix-based approach from [LAR$^+$19] is much higher. Even for the smallest benchmark (MBIST_1_5_5) from this set, the runtime is improved by a factor of larger than 5400x. The runtime and memory consumption of the matrix-based compliance analysis approach from [LAR+19] increases dramatically, as the size of RSN matrices rises for larger RSN designs. Moreover, the size and complexity of the ILP equations required for the resynthesis in the approach from [LAWW20] is significantly larger in comparison to the ILP equations required in the presented method. Therefore, for all other benchmarks from the DATE'19 benchmark set, the computation is performed with the presented improved approach only.

Reachability Evaluation

Explicit security specifications are modeled by a list of instruments pairs, where data propagation is prohibited between the instruments. The instrument pairs are built randomly, and the fraction of prohibited instrument pairs compared to the total number of instrument pairs varies from 0% to 100% with a step of 10%. For each given fraction of prohibited connectivities, the SCAR is performed to identify the violating hybrid paths. The number of security compliance violations is normalized against the total number of violations in a given RSN, and observed as the "fraction of violating connectivities". Fig. 6.13 shows the dependency between the fraction of prohibited connectivities between the system instruments and the average fraction of violations in the RSN.

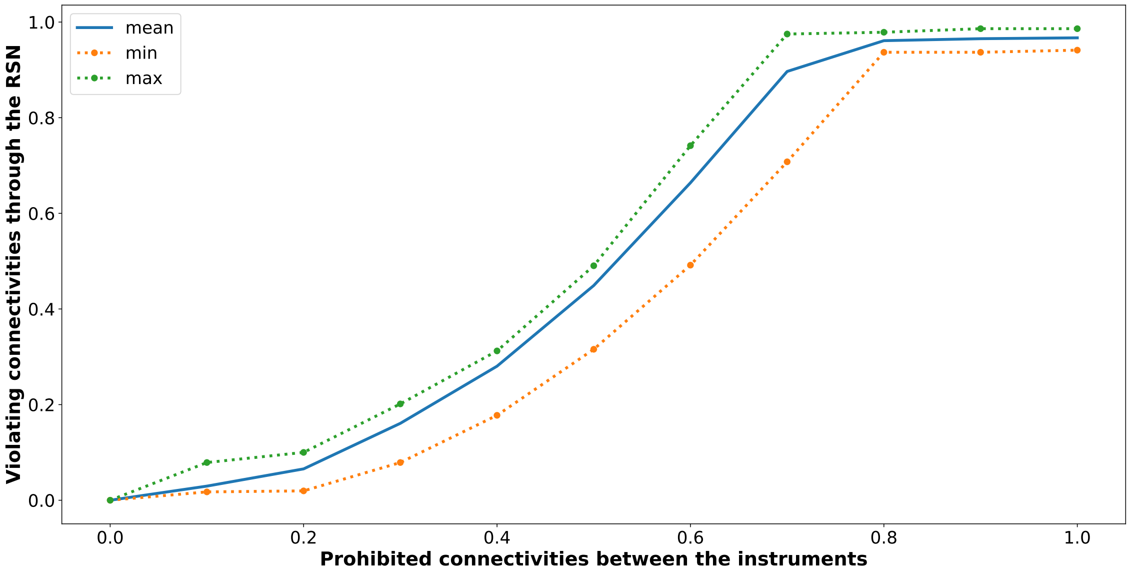


Figure 6.13: The reachability properties of the system and the RSN

The minimal and maximal fractions between the benchmarks are also shown in the diagram with a dashed line each. As the fraction of the prohibited connectivities between the instruments reaches 80 percent, the fraction of such connectivities in the RSN, which violate the compliance with the initial system, saturates for all the benchmarks.

Security Compliance Analysis and Resynthesis

The flow from Fig. 6.2 is performed to securely integrate an RSN into a given system. To assess the influence of the control signal assignments on the functional reachability, some correlation between the control signals has been added, such that 20% of the neighboring mux pairs are controlled by the same external control signals. Complementary to the implicit security specification of the connectivities between the instruments, an explicit security specification is defined by the designer: for 20% randomly selected instrument pairs, any data transfer is prohibited between the instruments through the RSN.

Fig. 6.14 shows the results of the security compliance analysis for the specification above. In particular, the relative numbers of structural and functional connectivities are given for all the benchmarks.

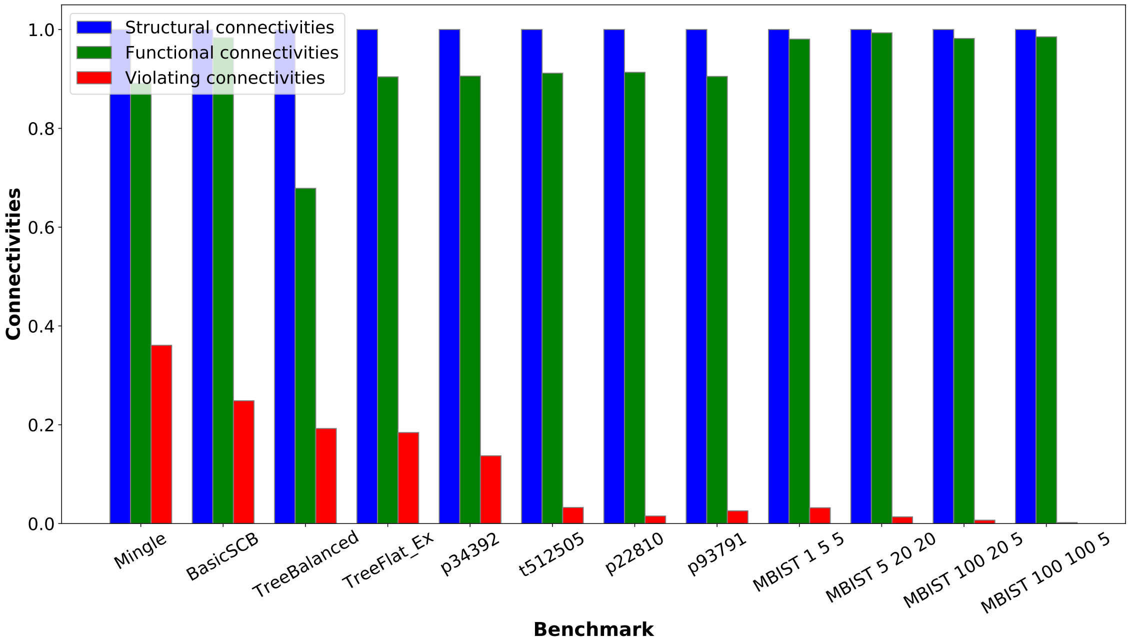


Figure 6.14: The connectivities through RSNs

Given the functional connectivities through an RSN considering retargeting, the connectivities between the instruments in the original system and the interconnections between the scan segments and the instruments, the violating connectivities due to improper RSN integration are calculated. All the numbers are normalized with respect to the total number of structural connectivities in a given benchmark. For absolute numbers, refer to Apppendix B.3.

All the identified violating connectivities have been resolved for all the benchmarks. In Fig. 6.15, the first bar for each benchmark shows a relative number of structural changes, which are required to resolve one security compliance violation.

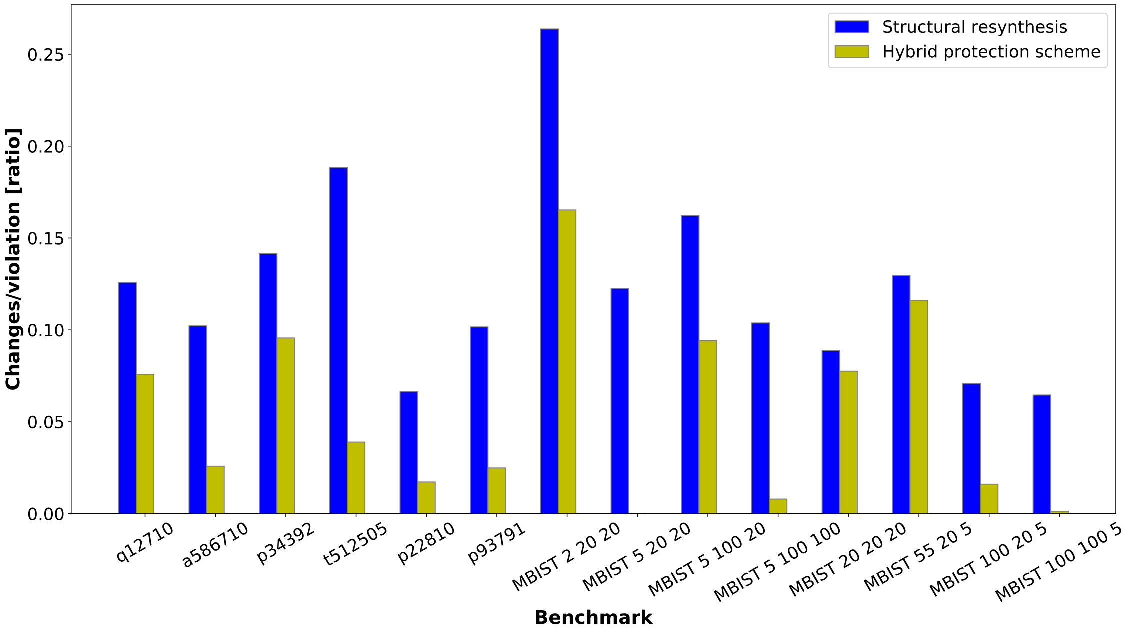


Figure 6.15: Relative number of changes per violation for the developed schemes

For all the benchmarks, the relative number of changes is significantly less than 1, which means that one structural change in average is able to resolve multiple violations.

6.3.2 Hybrid Protection Scheme for Multiple Users

Compared to the experimental setup from Section 6.3.1, two additional user groups have been considered besides the complete access for the manufacturer: one for the system integrators, and one for maintenance and user. The experiments deal with the general case of conflicting rights.

For all the benchmarks, all the violations have been resolved by applying a hybrid protection scheme. The security specification has been fulfilled. The developed filter applicability analysis has been performed to identify those violations which cannot be resolved by using a filter-based method without sacrificing the accessibility of some instruments via RSNs. The filter applicability fraction is measured for all the initial benchmarks as shown in Fig. 6.16.

The *filter applicability fraction* is a fraction of the number of violations, which are resolvable by a filter, $\#viol_{filter}$ compared to the total number of violations $\#viol$:

$$filter\ applicability\ fraction = \frac{\#viol_{filter}}{\#viol} \tag{6.12}$$

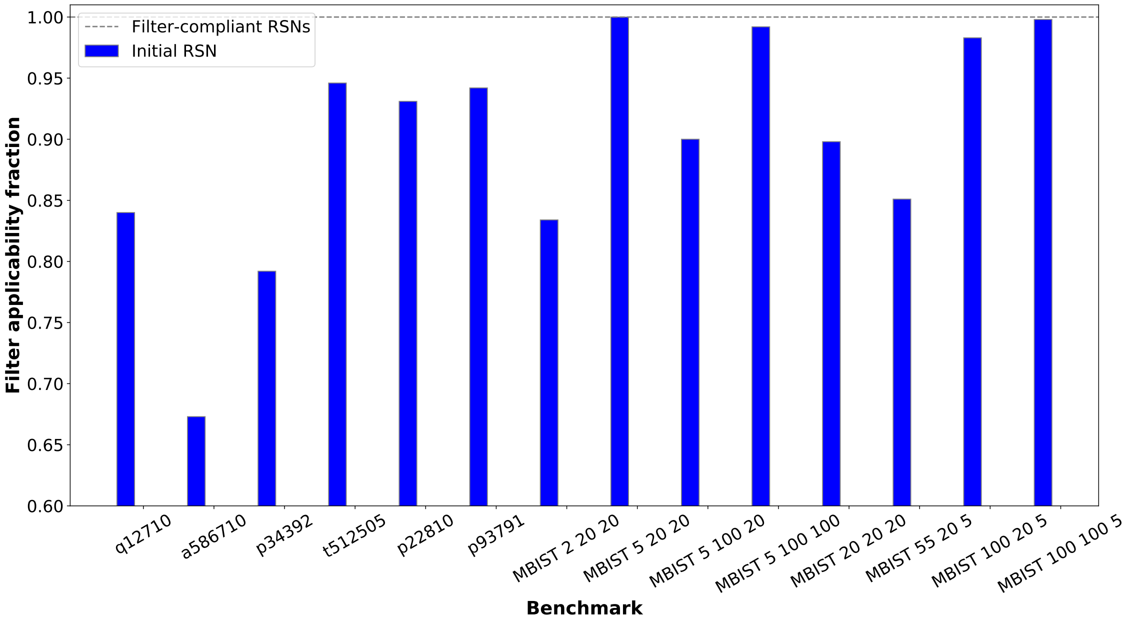


Figure 6.16: Filter applicability fraction of the initial RSNs

The filter applicability fraction shows that only for one benchmark (*MBIST 5 20 20*), applying a pure filter-based approach to resolve the violations would not make any required instrument inaccessible via the RSN. Therefore, to make the RSN filter-compliant, a few structural changes are still required by the hybrid protection scheme. The relative number of structural changes is given in the second bar of Fig. 6.15 for each benchmark. For all the benchmarks, compared to the pure structural solution, less hardware overhead is required. Since the majority of violations are resolved by a filter in a hybrid scheme, the number of required structural changes to the RSN is dramatically decreased. Moreover, the hybrid approach can specify different access rights for various user groups, which is not possible by using the structural resynthesis alone.

The scalable processing algorithm made it possible to keep the runtime is less than one hour even for the largest benchmarks. For most benchmarks, just a few minutes are required to perform the hybrid protection scheme.

6.4 Summary

To mitigate sensitive data leakage, a system designer may thoroughly develop the connectivities between the components in a security-aware manner. Test infrastructure may introduce additional unwanted connectivities into the initial system. These connectivities might be misused by an attacker to leak sensitive data. Even though test infrastructure poses a security threat, it cannot be disconnected from the system after the manufacturing, since the access to on-chip infrastructure should remain for the whole system lifecycle, e.g., for monitoring and maintenance.

The high sequential depth and the complex control dependencies of RSNs make their security-compliant integration extremely difficult. This chapter presents the first complete approach to integrating RSNs in a security-preserving manner, which is scalable with increasing RSN size and complexity. The developed scheme first analyzes the connectivities in the original system and after the RSN integration. The functional connectivities inside the RSN are computed considering the extra-dependencies, which arise due to retargeting. The developed scheme then detects all the unwanted extra-connectivities, which are introduced into the original system due to improper RSN integration.

Finally, the detected violating connectivities are efficiently resolved using the developed resynthesis scheme. First, a structural resynthesis scheme is developed, which physically resolves the violations between the instruments of the original system. Since all the violations are resolved simultaneously, the number of changes to the original RSN compared to the state-of-the-art methods is significantly reduced. Divide-and-conquer-based processing makes the developed method scalable for large and complex RSN designs. The structural resynthesis enables the integration of security-compliant RSNs for the case if all the user groups have the same access rights and the security specification does not change throughout the system's lifetime.

The method above also offers a solid background for an automated hybrid protection scheme for RSNs. This scheme combines the benefits of the structural resynthesis approaches with the flexibility of the functional filter-based methods, and overcomes their limitations. A minimized number of structural changes is applied to RSNs to preserve the accessibility of the instruments. The remaining violations are flexibly resolved by using a Finite State Machine-based filter. The resulting protection scheme considers complex security-preserving access scenarios for multiple user groups with specific permissions. The scheme does not require additional changes to the RSN structure, even if the security specification changes while preserving the accessibility of all required instruments through the RSN.

Chapter 7

Integration of Dependable RSNs

So far, a precise analysis of specific dependability properties of RSNs has been presented, followed by efficient methods to enhance these properties for a given RSN. At the same time, it is almost impossible and meaningless to isolate specific dependability properties from one another. To address this problem, the current chapter presents a complete flow to integrate RSNs in a dependable way, which considers such dependability properties as robustness, testability and security compliance of RSNs altogether.

7.1 Complete Solution for Resynthesis of Dependable RSNs

In Table 7.1, specific dependability properties, which have been discussed so far, are mapped to the required changes to RSNs.

Table 7.1: Influence of dependability enhancing resynthesis schemes on RSNs

Property	**Changes to an RSN**	**Ref.**
Security Compliance	Structural changes to the RSN topology	[LAR$^+$19, LAWW20]
Testability	Length of single segments Feedback loops for single segments	[LWW21, LWW22a] [UKW17, LWW22a]
Robustness	High-yield cells for certain scan primitives	[LWW22c]
ROSTI concurrent test	Access interface, no structural changes	[WLA$^+$21, LWW22a]
Online periodic test	New patterns, no structural changes	[LWW22b]
Access rights & remaining violations	Access interface, no structural changes	[LAW21]

The changes to the RSN topology are shown in yellow, while the changes to the segments' length are shown in green. The changes to the properties of scan primitives which do not affect the required access patterns are shown in blue. Finally, those enhancements which do not require any structural changes are shown in orange.

Example: *The running example is repeated in Fig. 7.1 to support better readability of the document. The security properties of the initial system are provided as connectivities between the instruments. The violating connectivities from the vertex* s_1 *to* s_3*; from* s_2 *to* s_4 *and from* s_3 *to* s_4 *are identified by the developed security compliance analysis (see Chapter 6.1). An access specification is formulated as follows (see Chapter 6):*

- *An access to a safety-critical instrument* i_3 *is required for all user groups.*
- *An access to* i_4 *is allowed for test engineers, but is restricted for regular users.*

In this chapter, the example is modified step-by-step to obtain a dependable RSN.

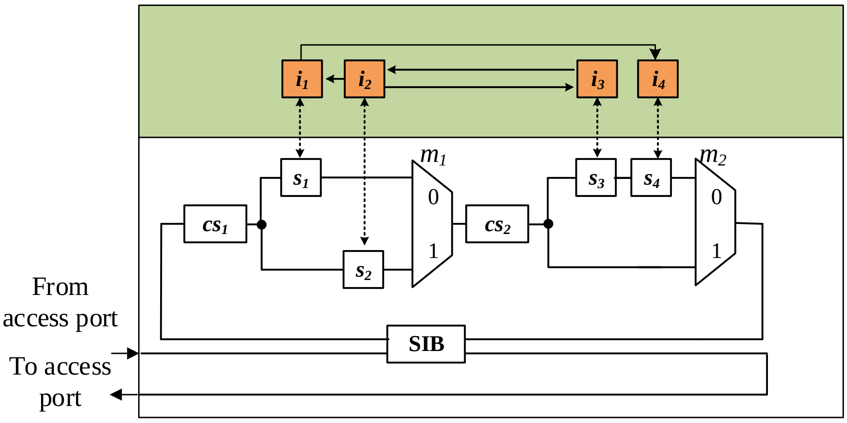


Figure 7.1: Initial RSN

The remainder of this chapter discusses the influence of the presented resynthesis schemes on the RSN structure. The required changes for each dependability property are analyzed and a complete solution is provided for the resynthesis of dependable RSNs. The developed solution avoids any contradictions between the considered dependability properties. The following steps with regard to the required changes are performed to obtain a dependable RSN.

7.1.1 Structural Changes to the RSN Topology

A precise security compliance analysis identifies the violating connectivities in the RSN. All the identified violations are classified with respect to the filter applicability. Those violations, which cannot be resolved by using a sequence filter, are handled by using a structural resynthesis scheme. After applying the resynthesis method, the existing violating connectivities are removed from the RSN. Some novel connectivities may be introduced into the RSN to ensure the accessibility of all the instruments via the RSN. A security compliance analysis and a filter applicability analysis are applied again to validate if all the remaining violations can be resolved by a filter. The structural resynthesis is applied until all the remaining violations are proved to be resolvable by using a filter without loosing access to the instruments. Here, such optimization criteria as access latency and hardware overhead of the RSN are considered to enhance the resynthesis procedure even more.

Result: *Security-enhancing resynthesis may require changing the functional connectivities inside the RSN.*

Example: *A Filter Applicability Analysis shows that only the violating connectivity between the segments* s_3 *and* s_4 *cannot be resolved by using a sequence filter. In Fig. 7.2, the structural resynthesis moves the scan segment* s_4 *to a separate branch of the scan multiplexer* m_2 *to resolve a security compliance violation between the scan segments* s_3 *and* s_4 *(see Chapter 6).*

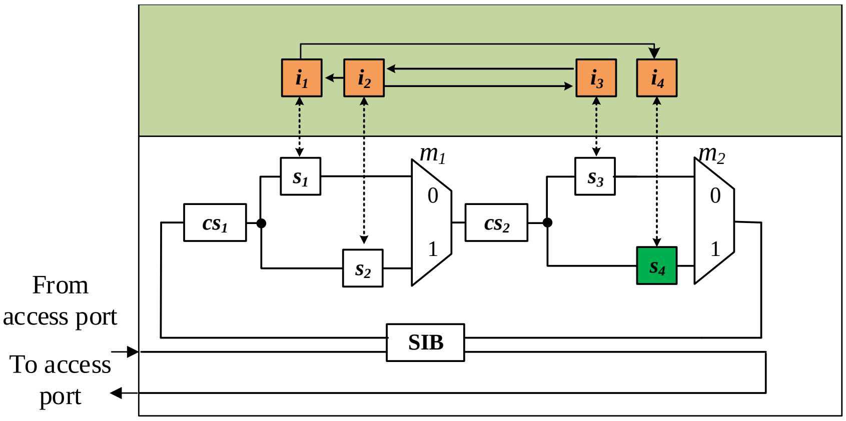


Figure 7.2: Structural changes to the RSN topology

7.1.2 Changes to the Segments Lengths

Testability analysis is applied to identify those spots in the RSN, which cannot be tested by an altered path length. The testability is enhanced with a minimized hardware overhead such that any fault in the control logic results in the altered active scan path length.

Result: *Since only some buffer scan cells are added into the initial RSN, the presented approach does not affect the functional connectivity and thereby also the security compliance [LAR+19] of the RSN with the underlying system.*

Example: *In Fig. 7.3, the lengths of the segments* s_1 *and* s_3 *are slightly increased to ensure the testability of control lines (see Chapter 5)*

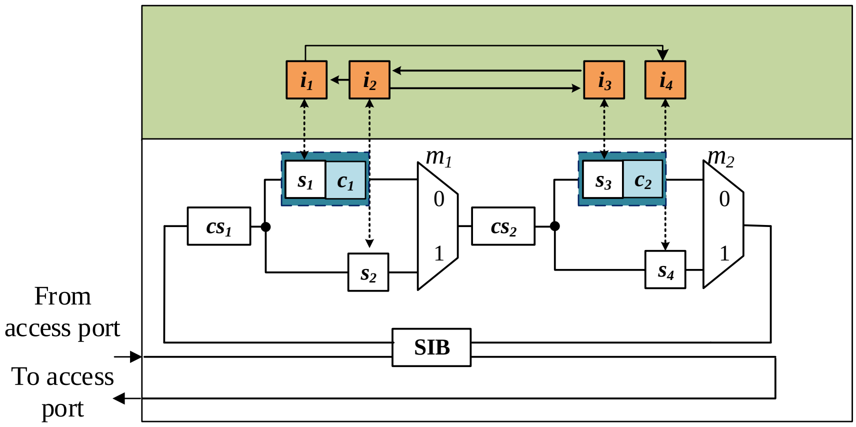


Figure 7.3: Changes to the segments lengths

7.1.3 Changes to the Properties of Scan Primitives

Testability of scan interfaces is enhanced by adding extra feedback loops into a scan segment. To ensure robust access to the instruments, a small number of scan primitives is selected based on a precise RSN criticality analysis and is hardened by using high yield cells. Here, a tradeoff between the remaining damage to the system in presence of the RSN defects and the cost of hardening is considered.

Result: *Each robust scan primitive with a testable scan interface can be represented as a black box, which has the same interfaces and the same timing, compared to the initial primitive. A top-level RSN structure is not affected by the changes of specific scan primitives. If the initial*

RSN is security compliant [LAWW20] and testable with respect to faults in the control logic [LWW21], the performed changes do not destroy these properties.

Example: *In Fig. 7.4, the control scan primitives in yellow are hardened to preserve a reliable access to a safety-critical instrument* i_3 *(see Chapter 4). The interfaces to the instruments are enhanced by injecting feedback lines (see Chapter 5).*

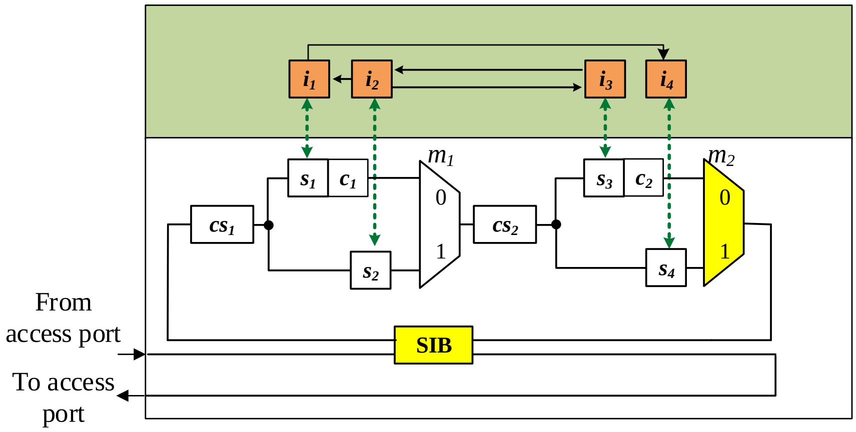


Figure 7.4: Changes to the properties of scan primitives

7.1.4 Dependable Access Interface

A security filter is integrated as a separate hardware block to resolve the remaining violations, which are identified by the security compliance analysis. Thanks to an accurate filter applicability analysis, it is ensured that the filter-based protection does not affect the authorized accessibility of the instruments. Using a security filter allows decreasing the hardware overhead of a security preserving RSN resynthesis significantly compared to a purely structural solution.

The introduced self-test scheme ROSTI tests the scan segments on the currently activated scan path prior to any access pattern. Applying the ROSTI-scheme reduces the test access time and does not require any changes to the RSN itself.

The solutions above can be combined into a dependable access interface for RSNs, which uses a conventional CSU-protocol and is placed directly at the primary scan-input of the underlying RSN. Such an interface is compliant with extensions of the RSN standards like the P1687.1 proposal [CVTR20] which defines access to RSNs through alternate interfaces.

An online periodic test avoids fault accumulation in rarely used scan primitives. The generated access patterns cover all the primitives with a minimized test cost. They can be stored on-chip and provided to the RSN via the dependable access interface above. The developed test method can be used throughout the RSN lifetime. The workload patterns can be efficiently generated by the existing test sequence generation methods [UKW17, KBSW16, CDRS20, CSSS18, HHD20].

Result: *The presented schemes do not affect the connectivities inside the RSN, the lengths of the paths or the properties of single scan primitives.*

Example: *In Fig. 7.5, the RSN is augmented with a dependable access mechanism, which includes both the secure filter and the ROSTI-self-test (see Chapter 6 and Chapter 5). Online periodic test patterns are uploaded into the chip.*

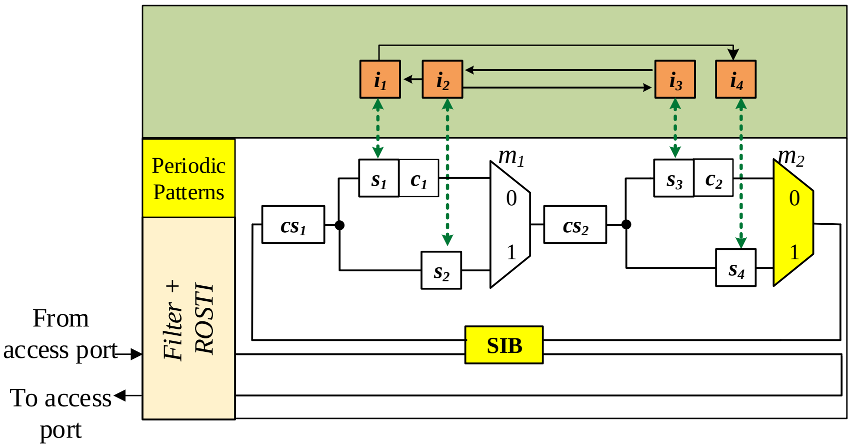


Figure 7.5: Dependable RSN

7.1.5 Compatibility of the Applied Changes

Assume that the changes to the RSN design and operation are applied in the order described in this chapter. In this case, each further change does not destroy the properties, which have been enhanced at the previous steps. Thereby, a complete methodology for integration of dependable RSNs is presented. It does not sacrify any dependability properties in order to enhance some other dependability aspects. By integrating a dependable RSN, many important optimization criteria, such as access latency, hardware overhead, robustness of an access, have been considered on different steps of the dependability-enhancing resynthesis scheme.

7.2 Summary

A complete integration methodology for dependable RSNs is presented. It combines precise analysis methods for individual dependability properties with efficient dependability-enhancing resynthesis of RSNs. Important dependability aspects are considered, such as robustness, testability, and security compliance of RSNs. The resulting RSNs support dependability management throughout the system lifecycle. In the presented solution, a common contradiction between the testability of the design with its security compliance is resolved for the case of Reconfigurable Scan Networks. Different levels of changes to the RSN structure are efficiently exploited to fully mitigate any conflicts between the dependability properties of RSNs. The presented scheme is extendable to other properties of RSNs, which are important for a DfT integrator.

Part III

Conclusions and Future Work

Chapter 8

Conclusions

8.1 General Conclusions

The design of dependable systems requires considering such dependability properties as security, reliability, and functional safety. Throughout the whole system's lifetime, the dependable operation of the system is supported by a high number of test and access insertions. These insertions perform post-silicon validation, manufacturing test and diagnosis, and in-field dependability management. The data for these insertions is collected by a vast amount of functional and non-functional instruments, which range from sensors and monitors, and BIST registers to runtime-adaptive instruments, such as Adaptive Voltage and Frequency Scaling and Adaptive Body Bias blocks.

Reconfigurable Scan Networks provide efficient and flexible access to the instruments. They support the dependability management throughout the system lifetime. RSNs are used to collect the evaluation results from the instruments and control their operation offline and in the field. Moreover, RSNs may guide the operation of fault-handling mechanisms in the case of performance degradation of specific modules, perform online health monitoring, and even be used to detect security attacks.

At the same time, if an RSN is integrated improperly, system dependability might be compromised. Even a single fault in an RSN might make certain parts of the RSN inaccessible, and, as a result, affect the accessibility of the instruments via the faulty RSN. If such a fault remains undetected by the existing test methods, data from a wrong instrument can be erroneously captured and silent data corruption may happen. Alternatively, control patterns for a runtime adaptive instrument might reach the wrong instrument and lead to an erroneous configuration of the system. Moreover, an unwanted extra-connectivity, which is introduced into the system due to improper RSN integration, might serve as a side-channel for an attacker to gain unauthorized access to system internals, which might include disabling such safety-critical components of the system as automotive breaks or leak sensitive data. Any of the dependability threats mentioned above might lead to a system failure or even result in permanent damage to the system.

In this document, an automated approach is presented to synthesize RSNs, such that the dependability threats due to improper RSN integration are mitigated. The integration scheme is based on a precise dependability analysis and an efficient dependability-enhancing resynthesis of RSNs. The resulting RSNs can be applied to support the system dependability management throughout the whole system lifecycle.

- *Robust access to the dependability instruments* via RSNs even in the presence of defects in RSNs is ensured by applying an efficient selective hardening scheme. The developed scheme is based on the precise criticality analysis of the RSN components.
- Even if the most critical primitives are hardened, the probability of a fault in an RSN cannot be neglected. The existing test methods do not guarantee complete coverage. To overcome this challenge, a complete design-for-test scheme presented in this document ensures *fault detection* in RSNs. Here, testability analysis is applied to accurately identify such faults which remain undetectable by the existing test methods. The testability of such faults is improved. Thereby, faults which might corrupt the correct activation of scan paths through the RSNs, communication with the instruments, as well as the integrity of the activated scan paths, are detectable. Additionally, a method for an online periodic test of RSNs is developed. It allows for mitigating fault accumulation in rarely used components and is applied complementary to the developed DfT scheme and the existing test generation methods.
- Finally, *unwanted extra connectivities*, which arise due to an improper RSN integration, are identified by the presented security compliance analysis scheme and mitigated by a cost-efficient resynthesis scheme. Different access rights for user groups, which might be necessary throughout the system lifecycle, are ensured by applying a

hybrid protection scheme, which combines pure structural resynthesis with dynamic security properties.

Dependability properties mentioned above are enhanced. The interdependencies between specific dependability properties are carefully analyzed. As a result, the testability, security compliance, and robustness of the resulting RSNs are not contradicting.

Experimental results have shown that the developed approach is highly scalable with the increasing size and complexity of RSNs. Efficient modeling of RSNs makes it possible to overcome the challenges which arise due to the high sequential depth of RSNs and their complex control dependencies. Divide-and-conquer-based methods split complex decision and optimization problems of RSN analysis and resynthesis into simplier problem instances. The analysis and resynthesis problems are solved within an acceptable runtime even for the largest RSNs. The resulting RSNs are efficient in terms of latency and hardware overhead. By applying multi-objective optimization algorithms close-to Pareto-optimal solutions are obtained considering multiple contradicting optimization criteria.

8.2 Ongoing and Future Work

An automated approach to integrating dependable RSNs provides a solid background for further investigation and enhancement of different dependability properties of RSNs throughout the lifetime:

- The methods to provide robust access to the instruments can be extended to provide fault-tolerant access via RSNs for the case of multiple faults. By exploiting hardware redundancies, it can be ensured that the maximized number of instruments remain accessible via faulty RSNs by applying one or multiple access patterns.

- The methods to ensure the security compliance of a given RSN with the underlying system can be applied in the domain of functional safety. Unsafe operations, which could result e.g. in an unexpected change from a functional mode to a test mode, or in the deactivation of safety-critical components as breaks, will be identified and an RSN will be resynthesized efficiently to mitigate such safety threats.

- Due to the enhanced testability of RSNs for the faults in the control logic and the scan interfaces, enhanced RSN test and diagnosis methods can be developed. The presented online test method can be used as a basis for an online diagnosis method.

To assess the quality of the provided test and diagnostic solutions and to validate the timing properties of the resulting RSNs throughout the lifetime, an efficient simulation platform is required. The platform should consider a realistic variation-aware model of a scan cell to identify possible timing violations, such as setup- and hold-time violations, and can be further used to support the dependability-enhancing RSN resynthesis in a way that a sweet spot between the frequency of an RSN and its influence on the integrity of the data in the underlying system is considered.

Appendices

Chapter A

Experimental Setup

This section briefly discusses the experimental setup for all the experiments. All the experiments have been conducted on an Intel(R) Xeon(R) W-2125 CPU at 4.00GHz with 132 GB of main memory. The investigated benchmarks are presented in Section A.1. The information about the electronic design automation and other tools, which have been used to enable dependability-aware RSN integration, is presented in Section A.2.

A.1 Benchmarks

A.1.1 RSN Benchmarks

The RSN benchmarks have been taken from the commonly recognized ITC'2016 [TJD$^+$16], DATE'2019 [RTB$^+$19] and ITC'2002-based [BKW15b] benchmark sets. For the considered benchmarks the number of scan multiplexers (Column 2), SIBs (Column 3), scan segments (Column 4), scan cells (Column 5) and the highest hierarchy level (Column 6) are given in Table A.1.

A.1.2 Connectivities between the Instruments

The ISCAS'89 benchmarks [BBK89] represent the connectivities **between the embedded instruments** in the underlying system, which are accessed via RSNs. These are not all the connectivities in the system. These benchmarks have been used instead of pseudo-random connectivities between the instruments to avoid possible bias in the results.

Table A.1: Characteristics of benchmarks

(1)**Design**	(2) #muxes	(3) #sibs	(4) #segs	(5) #cells	(6) #lvl
ITC'16 Benchmarks					
BasicSCB	10	-	21	176	4
Mingle	13	10	22	270	3
TreeFlat	24	12	24	101	2
TreeUnbalanced	28	28	63	41,887	11
TreeBalanced	46	43	90	5,581	7
TreeFlat_Ex	60	57	123	5,194	5
q12710	25	25	47	26,183	2
a586710	47	-	79	41,682	3
p34392	142	-	245	23,261	3
t512505	160	-	288	77,006	2
p22810	283	283	537	30,111	3
p93791	653	-	1,241	98,637	3
DATE'19 Benchmarks					
MBIST_1_5_5	15	8	113	548	4
MBIST_1_5_20	15	8	338	1,523	4
MBIST_1_20_20	45	23	4,179	6,068	4
MBIST_2_5_5	28	16	224	1,091	4
MBIST_2_5_20	28	16	674	3,041	4
MBIST_2_20_20	88	46	2,624	12,131	4
MBIST_5_5_5	67	40	557	2,720	4
MBIST_5_20_20	217	115	6,557	30,320	4
MBIST_5_100_20	1,017	515	32,557	151,520	4
MBIST_5_100_100	1,017	515	151,135	671,520	4
MBIST_20_20_20	862	460	26,222	121,265	4
MBIST_55_20_5	2,367	1,265	22,607	118 970	4
MBIST_100_20_5	8,102	2,300	41,102	216,305	4
MBIST_100_100_5	20,102	10,300	172,700	1,080,305	4
ITC'02 Benchmarks					
u226	59	-	99	1,457	2
d281	67	-	117	3,880	2
d695	178	-	335	8,407	2
h953	63	-	109	5,649	2
g1023	94	-	159	5,400	2
f2126	45	-	81	15,834	2
q12710	30	-	51	26,188	2
p34392	142	-	245	23,261	3

A.2 Electronic Design Automation (EDA) and other Tools

The experimental results have been obtained with the help of the following tools and frameworks:

- *EDA1687 tool*: The in-house EDA tool has been first introduced in [BKW15b] to model an RSN as a directed graph and to represent the data and the control dependencies between the scan primitives. It is written in C++ programming language. In this document, the tool has been significantly extended to implement the dependability-enhancing algorithms for RSNs, which are presented above in Part II.
- *ANTLR*: The parser ANTLR [Par13] has been used to build a graph model of an RSN from an Instrument Connectivity Language (ICL) description.
- *MiniSAT* The SAT-solver MiniSAT [ES04] has been used to support solving Boolean Satisfiability problem as a part of dependability analysis.
- *Gurobi*: The Integer Linear Programming (ILP) framework Gurobi [Gur19] has been used to support solving optimization problems as a part of the presented dependability-enhancing resynthesis algorithms.
- *Opt4J*: The framework Opt4J [LGRT11] has been developed by the group of Prof. Teich from Friedrich-Alexander-Universität Erlangen-Nürnberg. The SPEA-2 [ZLT01] evolutionary algorithm implemented in this framework has been used to support the multi-objective optimization algorithms for enhancing the dependability properties of RSNs.

Chapter B

Dependability Enhancement Results

This chapter summarizes the experimental results for the methods and algorithms presented above. The remainder of this chapter is split into three main sections as follows:

Section B.1 (Robust RSNs) provides the experimental results for the initial criticality analysis of scan primitives, followed by the robustness enhancing resynthesis method for RSNs.

Section B.2 (Testable RSNs) summarizes the experimental results for the complete design-for-test scheme for RSNs, which includes the methods for scan segments, scan interfaces and control lines, as well as the details about the test integration. The same chapter presents the experimental results for the developed online periodic test method for RSNs.

Section B.3 (Security Compliant RSNs) presents the results for security preserving RSN integration. It includes the security compliance analysis, as well as two security-enhancing resynthesis methods: for a single user and for multiple users.

B.1 Robust RSNs

The criticality analysis has been conducted considering an explicit specification, and the criticality of RSN primitives has been assessed. In the specification, 70% of all the instruments have randomly assigned non-zero damage weights of losing their observability, and another 70% - of losing their settability. The overlap part defines two non-zero weights.

Additionally, 10% random instruments have been set as important for observation, another 10% - for control.

The primitives to harden are selected by using the evolutionary algorithm called SPEA-2 [ZLT01] implemented in the Opt4J framework from [LGRT11]. The parameters below have been used for the optimization:

- Considered constraints: hardware costs and damage for the system operation;
- Size of the population: 300 for the benchmarks with more than 100 muxes, 100 for other benchmarks;
- Independent bit mutation probability: 0.01;
- Standard one-point crossover probability: 0.95.

The experimental results for all the considered benchmarks are provided in Table B.1. In the defect-free case, all the instruments are accessible. The information about the values of a cost function with respect to hardware cost and the resulting damage to the system is presented for three cases. First, the initial assessment is provided for the maximum cost of hardening, if all the primitives are hardened (Column 4). Also, the maximum damage in presence of single defects is provided, when none of the primitives is hardened (Column 5). The number of generations of an evolutionary algorithm is provided in Column 6.

The results of applying an evolutionary algorithm are provided in Table B.2. Next, the damage and the hardware cost are provided for two pareto solutions:

- The best damage-reducing solution, which requires at most 10% hardened primitives, in Columns 2 and 3.
- The most cost-efficient solution for reducing the damage down to 10% of the initially assessed value (Column 5 in Table B.1) in Columns 4 and 5.

All the important instruments remain accessible via the resulting RSNs. The runtime is provided in Column 6 and is acceptable for all the benchmarks.

Table B.1: Robust RSN Synthesis, Initial Assessment

	Benchmark characteristics		Initial assessment		
(1) Design	(2) # Seg.	(3) # Mux.	(4) Cost	(5) Damage	(6) # Gens.
BasicSCB	21	10	62	498	350
Mingle	22	13	56	373	300
TreeFlat	24	24	350	502	300
TreeUnbalanced	63	28	142	1,656	300
TreeBalanced	90	46	211	4,206	1,000
TreeFlat_Ex	123	60	289	597	2,000
q12710	47	25	127	576	300
a586710	79	47	155	1,010	2,000
p34392	245	142	482	7,932	700
t512505	288	160	713	7,146	1,000
p22810	537	283	1,298	22,911	1,000
p93791	1,241	653	2,946	293,771	3,500
MBIST_1_5_5	113	15	137	74,004	300
MBIST_1_5_20	1,523	15	362	632,421	400
MBIST_1_20_20	6,068	45	1,412	8,252,305	500
MBIST_2_5_5	1,091	28	137	83,509	500
MBIST_2_5_20	3,041	28	362	560,484	700
MBIST_2_20_20	12,131	88	1,412	8,174,778	700
MBIST_5_5_5	2,720	67	411	148,811	500
MBIST_5_20_20	30,320	217	385	6,175,005	900
MBIST_5_100_20	151,520	1,017	7,012	203,302,366	200
MBIST_5_100_100	671,520	1,017	93,447	2,138,755,955	1,500
MBIST_20_20_20	121,265	862	1,412	6,175,005	900
MBIST_55_20_5	216,305	8,102	512	814,369	500
MBIST_100_20_5	118,970	2,367	512	639,278	1,800
MBIST_100_100_5	1,080,305	20,102	2,512	20,977,832	1,200

B.2 Testable RSNs

In this section, the experimental data for the developed testability enhancing methods is given. First, more experimental results are provided for the testability-enhancing resynthesis of RSNs, which ensures the detection of all single faults in the control logic. Next, the experimental data for test integration of the complete DfT scheme for RSNs is presented. Finally, the data for the developed online periodic test of RSN is provided.

B.2.1 Control Lines

As discussed in Section 5.3.1, for the most benchmarks from the popular benchmark sets, the testability property has been proven. Therefore, the applicability of the structural

Table B.2: Robust RSN Synthesis, SPEA-II, Varying Optimization Criteria

	Min. cost, Damage $\leq 10\%$		Min. damage, Cost $\leq 10\%$		Runtime
(1) Design	(2) Cost	(3) Damage	(4) Cost	(5) Damage	(6) [m:s]
BasicSCB	5	50	6	30	00:01
Mingle	4	33	5	20	00:01
TreeFlat	7	42	8	26	00:07
TreeUnbalanced	10	155	14	31	00:02
TreeBalanced	18	362	21	216	00:03
TreeFlat_Ex	29	57	28	60	00:04
q12710	8	27	12	19	00:03
a586710	5	90	15	24	00:15
p34392	8	683	48	68	00:34
t512505	21	699	71	121	00:16
p22810	33	2,215	28	3,712	01:01
p93791	38	28,681	286	561	06:10
MBIST_1_5_5	32	7,176	13	20,799	00:26
MBIST_1_5_20	35	62,264	36	60,344	02:21
MBIST_1_20_20	129	801,889	137	752,261	10:01
MBIST_2_5_5	19	8,141	13	12,081	03:45
MBIST_2_5_20	34	54,314	36	50,060	04:17
MBIST_2_20_20	129	788,085	138	722,191	08:18
MBIST_5_5_5	8	14,213	41	163	01:10
MBIST_5_20_20	127	614,605	36	1,343,502	15:02
MBIST_5_100_20	1,983	20,555,328	701	48,147,171	35:17
MBIST_5_100_100	17,066	213,650,290	8,625	405,742,391	92:01
MBIST_20_20_20	131	605,065	141	537,474	23:40
MBIST_55_20_5	112	78,595	51	208,782	05:43
MBIST_100_20_5	87	63,268	51	144,057	07:15
MBIST_100_100_5	273	2,096,139	248	2,396,324	59:32

resynthesis on these benchmark sets can only be validated on a limited number of the initial benchmarks. To create a representative benchmark set, which contains a high number of large RSN designs, additional bypass registers with a controllable length are implemented. The testability of the modified benchmarks has been analyzed and the benchmarks have been resynthesized to ensure the unique path lengths within a single fault assumption in the switch logic.

The results are presented in Table B.3. Column 2 shows a number of additional scan cells, which have been added into the RSN. In average, one additional scan cell has been required for each modified scan segment. Compared with the total number of scan segments in RSNs (Column 4, Table A.1), and especially with the number of scan cells (Column 5, Table A.1), the presented method requires a negligible hardware overhead, and the in-

crease of the instruments access latency through the resynthesized RSN is negligible as well. The number of virtual changes, which have been performed to transform an arbitrary graph into a series-parallel form, is given in Column 3 for all the benchmarks.
The number of faults in the control scan primitives is given in Column 4 for all the benchmarks, and includes all single stuck-at faults, affecting control primitives. In Column 5, the number of undetected faults, whose detection relies on ASP length, $UDT-PL$ is given for the initial RSNs. For the resynthesized RSNs, after the presented method is applied, each single fault in the control logic is detectable by an altered path length, as validated by the repeated testability analysis. The runtime of the approach is acceptable and requires less than 25 minutes for the largest benchmarks, and just few seconds for the most of the benchmarks (Column 6).

B.2.2 Test Integration

In this section, the complete developed DfT approach is evaluated. The experimental results for the developed scheme are shown in Table B.4:

- *Integration of the Design-for-Test scheme:*
    - *Scan interfaces:* Scan registers are enhanced by injecting a feedback line.
    - *Control lines:* The testability with respect to single faults in the control logic is proven for all the benchmarks. The total number of faults in control lines is given in Column 2. The runtime is provided in Column 3 and is negligible for all the benchmarks.
    - *Scan segments:* ROSTI is integrated to generate self-test for scan segments.
- *Simulation of test sequences:*
    - *Scan interfaces:* A test sequence is generated to test scan interfaces in the enhanced RSN. The test cost is provided in Column 4.
    - *Scan segments:* To test scan segments, an access sequence is constructed of a workload sequence for testing the interfaces and a flush test sequence. The workload sequence is provided by an external tester. It ensures that the scan primitives of an RSN are covered by activating a minimized number of activated

Table B.3: Testability-oriented resynthesis to detect all UDT-PL

(1) **Design**	(2) #cells	(3) #virtual	(4) #fault	(5) UDT-PL	(6) time[s]
BasicSCB	8	0	40	16	1.0
Mingle	4	0	52	8	1.2
TreeFlat	2	2,047	48	4	25.5
TreeUnbalanced	9	0	112	18	2.1
TreeBalanced	9	0	200	22	2.4
TreeFlat_Ex	28	0	256	56	3.5
q12710	23	0	108	46	1.0
a586710	22	0	128	44	1.3
p34392	73	0	388	146	2.2
t512505	107	0	636	214	8.6
p22810	224	0	1,080	460	3.2
p93791	550	0	2,384	1,100	10.1
MBIST_1_5_5	4	2	30	10	1.1
MBIST_1_5_20	4	2	30	8	1.6
MBIST_1_20_20	15	2	90	30	3.3
MBIST_2_5_5	7	2	56	14	2.2
MBIST_2_5_20	9	2	56	18	0.8
MBIST_2_20_20	32	2	176	64	9.1
MBIST_5_5_5	20	2	134	40	1.1
MBIST_5_20_20	75	2	434	150	26.1
MBIST_5_100_20	365	2	2,034	730	120.7
MBIST_5_100_100	375	2	2,034	750	1453.0
MBIST_20_20_20	300	2	1,724	600	21.4
MBIST_55_20_5	866	2	4,734	1,732	356.2
MBIST_100_20_5	1,586	2	8,604	3,172	168.1
MBIST_100_100_5	7,600	2	40,204	15,200	280.3

scan paths as in [BKW15b]. The flush test sequences to test the shift logic are generated by ROSTI. The test cost is given in Column 5.

The test cost using the developed test integration is compared with the application of the test sequences for scan interfaces and scan segments one after another. The test cost is given in Column 6 and is calculated as the sum of the individual test costs in Column 4 and Column 5.

- *Test cost reduction:* The test sequences of the scan interfaces are now integrated into the workload, and a flush sequence tests the shift logic. The length of an activated path is used as an indicator for faults in the control logic. Column 7 provides the test cost for the case when the developed test integration scheme is considered and the bit-sharing mechanism is activated. This value provides a significant improvement compared to testing different fault locations individually (Column 6).

Table B.4: Design-for-Test Scheme Results: Test Cost

	Control lines	Runtime	Test Cost [#cycles]			
(1)Design	(2) #faults	(3) [s]	(4) Inter.	(5) Segs.	(6) Sum	(7) Our
u226	118	0.2	15,536	22,647	38,183	17,996
d281	134	0.2	32,863	34,113	66,976	35,959
d695	356	0.3	84,026	116,234	200,260	93,566
h953	126	0.2	44,296	38,247	82,543	47,104
g1023	188	0.3	46,443	50,418	96,861	50,463
f2126	90	0.1	114,563	75,269	189,832	116,723
q12710	60	1.0	184,971	109,904	294,875	192,231
p34392	288	2.2	181,591	156,403	337,994	188.851

Area overhead compared to the underlying RSN is given in Column 1 of Table B.5 and is negligible. The fault coverage for RSN benchmarks without feedback lines in the scan segments is given in Column 2 of Table B.5. Fault sampling is used to p93791, t512505, and a586710 with a confidence interval of 99%. Fault coverage is 94.72% on average. To mitigate the coverage gap above, it is necessary to test the interfaces to instruments and logic. If scan segments are enhanced by integrating a feedback line and the workload patterns are used to test scan interface, a *complete fault coverage* is obtained for all the benchmarks.

In the resulting RSNs, faults in scan interfaces, control lines and scan segments are detectable. The scalability and effectiveness of the developed DfT scheme has been shown for a wide range of benchmarks.

B.2.3 Online Periodic Test

This section provides the experimental results for the developed online periodic test method. The test access sequence sets for performing the online periodic test of RSNs have been generated according to the method from Section 5.2. The coverage of scan

Table B.5: Design-for-Test Scheme Results: Overhead and Coverage

	Overhead[%]	Coverage [%]
(1)Design	(2) w.r.t. RSN	(3) w/o feedback
u226	0.69	93.65
d281	0.27	95.51
d695	0.19	92.17
h953	0.18	96.32
g1023	0.20	95.66
f2126	0.06	95.11
q12710	0.03	95.43
p34392	0.06	94.70

primitives is defined as a fraction of scan primitives, which are accessed by a given test sequence set, from the whole set of scan primitives. The solution space has been explored and a tradeoff between the test cost and the coverage of scan primitives has been investigated. A set of close to pareto-optimal solutions has been generated by means of genetic algorithms. For a required coverage, the best generated solution with respect to test cost has been selected. The genetic programming solver Opt4J [LGRT11] is used with the NSGA-II method [DPAM02] to perform the optimization. The genotype describes a test sequence set as a sequence of Boolean values, where each sequence comes after another, and each sequence contains one or multiple activated paths. Each bit shows the state of a control scan primitive on the path. The initial genotypes are created following Algorithm 1. Each next path is chosen so that the gain is maximized as shown in Section 5.2.3. The choice is randomized within the most prominent candidates to efficiently explore relevant parts of the search space.

The crossover operation between two-parent genotypes creates two child genotypes from two parental genotypes. The first child inherits the first part of test sequences from the first parent and the second part from the second parent, for the second child the inheritance is reversed. The mutation operation randomly removes one sequence from the test sequence set or adds a new sequence into the set.

The parameters below have been used for the optimization:

- Considered constraints: test cost and coverage;

- Standard one-point crossover probability: 0.95;
- Independent bit mutation probability: 0.05;

The exact setup for individual benchmarks is given in Table B.6. The number of generations to find a solution is given in Column 2 and is determined as a minimum number of generations when the solutions stop improving for 5 consequent operations. The size of the initial population is given in Column 3.

Table B.6: Periodic Test of RSNs: Genetic Algorithm

	NSGA-II	
(1)**Design**	(2) # generations	(3) # population
BasicSCB	10	100
Mingle	30	300
TreeFlat	10	100
TreeUnbalanced	30	300
TreeBalanced	30	300
TreeFlat_Ex	30	300
q12710	10	300
a586710	30	300
p34392	30	500
t512505	50	1000
p22810	100	1000
p93791	100	1000

The details about the generated solutions are given in Table B.7. The time to generate the solution is provided in Column 2, which is rather low for all the benchmarks. Column 3 shows the number of required test sequences and Column 4 represents the total number of configurations.

For all the benchmarks, the generated test sequence set covers all the scan primitives. The information about the test cost of the generated test sequence sets is given in Table B.8. The test cost for the full coverage of scan primitives is given in Column 2. Given a minimal required coverage of scan primitives, e.g. 90%, the genetic algorithm generates a test sequence set with a significantly decreased test cost, as provided in Column 3.

Table B.7: Periodic Test of RSNs: Sequence Optimization Results

(1)**Design**	(2) runtime [m:s]	(3) # sequences	(4) # configurations
BasicSCB	00:32	1	7
Mingle	02:11	1	11
TreeFlat	00:02	2	4
TreeUnbalanced	02:16	7	39
TreeBalanced	05:02	2	9
TreeFlat_Ex	04:11	2	9
q12710	01:12	1	4
a586710	04:30	1	7
p34392	10:11	3	2
t512505	05:16	2	6
p22810	10:10	9	41
p93791	15:13	4	28

B.3 Security Compliant RSNs

This section presents the experimental results, which show the efficiency and the scalability of the presented scheme to integrate security compliant RSNs. First, the experimental results for the structural resynthesis method. Next, the results of the hybrid protection scheme for multiple users are presented.

B.3.1 Structural Resynthesis for Single User Protection

The flow from Fig. 6.2 is performed to securely integrate an RSN into a given system. To assess the influence of the control signal assignments on the functional reachability, some correlation between the control signals has been added, such that 20% of the neighboring mux pairs are controlled by the same external control signals. Complementary to the implicit security specification of the connectivities between the instruments, an explicit security specification is defined by the designer: for 20% randomly selected instrument pairs, any data transfer is prohibited between the instruments through the RSN.

The remainder of this subsection presents the results for the security compliance analysis and resynthesis schemes. The experimental results for the developed security compliance analysis are shown in Table B.9:

Table B.8: Periodic Test of RSNs: Coverage

(1)**Design**	(2) Coverage 100%	(3) Coverage 90%
BasicSCB	265	102
Mingle	385	199
TreeFlat	3,890	2,506
TreeUnbalanced	66,416	28,464
TreeBalanced	57,610	31,747
TreeFlat_Ex	24,226	11,078
q12710	39,394	30,118
a586710	55,957	34,376
p34392	40,923	20,410
t512505	143,112	102,557
p22810	379,373	210,214
p93791	1,225,512	469,840

- Firstly, the structural connectivities (Column 2, $\#struct.$), as well as the valid connectivities for individual ASPs (Column 3, $\#ASP$), and the functional connectivities (Column 4, $\#func.$) inside the RSN have been computed.

- Secondly, all the security compliance violations (Column 5, $\#viol.$) have been identified.

Table B.10 presents the experimental results for the developed structural resynthesis method. All the violations have been resolved with a few iterations of the flow from Fig. 6.2 (Column 2, $\#iter$), by removing just a few edges (Column 3, $\#removed$). The accessibility has been reintroduced by adding at most $2 \times \#removed$ edges into the graph. The minimized access latency ("reduce latency") and the minimized hardware overhead ("reduce overhead") have been used as the optimization criteria. The actual values of the average access latency and the hardware overhead have been measured for both optimization criteria. The values of these metrics for the resulting RSNs are normalized with respect to the values obtained for the initial RSNs and are shown in Columns 4, 5, 6, 7.

- When applying the "reduce latency" criteria, the resulting access latency of the segments was reduced as expected, and the hardware overhead increased with respect to the original RSN. This means that in the resulting RSNs, a larger number of shorter

Table B.9: Security Compliance Analysis

(1) Design	(2) $\#struct.$	(3) $\#ASP$	(4) $\#func.$	(5) $\#viol.$
BasicSCB	181	175	178	45
Mingle	230	155	205	83
TreeFlat	300	263	300	103
TreeUnbalanced	2,016	1,820	1,987	340
TreeBalanced	4,272	2,688	2,899	822
TreeFlat_Ex	7,869	6,987	7,116	1450
q12710	1,275	1,020	1,219	614
a586710	1,430	1,034	1,599	345
p34392	15,937	12,122	14,435	2,187
t512505	41,328	30,623	37,677	1,346
p22810	137,550	97,731	125,637	2,104
p93791	721,269	523,454	652,825	18,610
MBIST_1_5_5	30,193	29,600	29,605	967
MBIST_1_5_20	231,043	222,124	229,475	365
MBIST_1_20_20	929,218	916,123	923,015	356
MBIST_2_5_5	60,327	57,122	59,153	1,056
MBIST_2_5_20	462,027	422,173	458,903	5,591
MBIST_2_20_20	1,858,377	1,804,592	1,845,983	3,437
MBIST_5_5_5	150,783	142,861	147,866	23,177
MBIST_5_20_20	4,645,908	4,124,739	4,614,941	63,284
MBIST_5_100_20	23,947,908	22,112,834	23,793,341	56,452
MBIST_5_100_100	452,167,908	423,175,253	451,493,341	54,765
MBIST_20_20_20	18,584,778	15,267,365	18460946	23,683
MBIST_55_20_5	6,929,683	6,605,819	6,803,666	17,349
MBIST_100_20_5	12,619.618	11,859.933	12.390,506	87,275
MBIST_100_100_5	77,299,618	75,738,342	76,158,506	154,235

scan paths is synthesized, which allowed the approach to mitigate information leakage through RSNs.

- For the "reduce overhead" criteria, the latency was slightly increased, while the hardware overhead either increased only negligibly or even slightly decreased.

  That means that in the resulting RSNs it was possible to prevent all security violations while preserving almost the same hardware overhead.

Runtime is negligible for all the considered benchmarks, and the whole flow requires less than 3.5 minutes for the largest benchmark, while the average runtime is about one minute. On average 33 percent of the runtime has been spent for the analysis, while the remaining time has been spent for performing the automated resynthesis.

B.3.2 Hybrid Protection Scheme for Multiple Users

Compared to the experimental setup from Section B.3.1, two additional user groups have been considered besides the complete access for the manufacturer: one for the system integrators, and one for maintenance and user. The experiments deal with the general case of conflicting rights. The implicit security specification is given as a list of connectivities between the instruments. The previously developed security compliance analysis method is used to identify the violating connectivities due to improper RSN integration.

Table B.11 provides the experimental results for the hybrid protection scheme. For all the benchmarks, the number of violations is given in Column 2. After a few of iterations of the "Prepare the RSN" step (Column 3), a filter-compliant RSN is obtained. Just a minor number of structural changes was required for all the considered benchmarks to transform the RSN into a filter-compliant RSN. The remaining violations are resolved using a filter. In the resulting RSNs, all the violations have been resolved, the security specifications are fulfilled, and the accessibility of the required instruments through the RSNs is preserved.

The area overhead (Column 4) of the generated filter is acceptable for all the RSNs. In order to preserve an acceptable complexity of a secure filter for the hierarchically organized DATE benchmarks, the filter is also constructed hierarchically out of multiple FSMs. Each FSM corresponds to a sub-RSN, which is used to access a part of the instruments, such as the memory BIST registers. As the size and the complexity of the benchmark increases, the relative area overhead decreases and becomes negligible for the largest RSNs. The runtime (Columns 5 and 6) is acceptable even for the largest benchmarks.

Comparison to the State of the Art

Filters: In contrast to the pure filter-based approach, as in [AKS^{+}18] all the violations have been resolved without sacrificing the instruments' accessibility. The filter applicability fraction (Column 3 in Table B.12) shows that only for one benchmark, applying a pure filter-based approach to resolve the violations would not make any required instrument inaccessible via the RSN.

Resynthesis: Compared to the number of structural changes required to solve all the identified security compliance violation by using the pure structural solution [LAWW20] (Column 2 in Table B.13), less hardware overhead is required (Column 3). Since the majority of violations is resolved by a filter in a hybrid scheme, the number of required structural changes to the RSN is dramatically decreased. Moreover, the hybrid approach is able to specify different access rights for various user groups, which is not possible by using resynthesis alone.

Table B.10: Security Compliant RSN Integration

	Resolve		**Reduce Latency**		**Reduce Overhead**	
(1) Design	(2)#*iter*	(3) #*removed*	(4) *latency*	(5) *HW*	(6) *latency*	(7) *HW*
BasicSCB	1	8	0.91	1.09	1.01	0.93
Mingle	1	9	0.90	1.02	0.97	0.93
TreeFlat	1	9	0.90	1.05	0.98	0.93
TreeUnbalanced	1	9	0.99	0.96	0.99	1.01
TreeBalanced	1	18	0.98	1.02	0.97	0.94
TreeFlat_Ex	1	20	0.76	0.97	1.15	0.85
q12710	1	22	0.72	1.21	1.13	0.80
a586710	2	37	0.63	1.67	1.12	0.97
p34392	3	21	0.75	1.49	1.12	0.96
t512505	2	36	0.66	1.61	1.16	0.93
p22810	1	127	0.78	1.60	1.10	1.01
p93791	3	463	0.59	1.72	1.18	0.85
MBIST_1_5_5	1	101	0.18	4.00	1.34	1.00
MBIST_1_5_20	1	52	0.28	3.24	1.56	1.01
MBIST_1_20_20	1	48	0.24	3.33	1.12	0.97
MBIST_2_5_5	1	104	0.29	2.12	1.12	0.98
MBIST_2_5_20	2	187	0.31	2.21	1.15	1.10
MBIST_2_20_20	2	358	0.49	1.99	1.15	1.02
MBIST_5_5_5	1	4,826	0.22	3.55	1.56	1.01
MBIST_5_20_20	1	6,712	0.19	4.05	1.23	0.97
MBIST_5_100_20	1	2,561	0.45	2.13	1.12	0.99
MBIST_5_100_100	1	972	0.65	1.16	1.18	1.09
MBIST_20_20_20	1	1,835	0.49	1.67	1.42	0.99
MBIST_55_20_5	2	1,240	0.69	1.46	1.14	0.98
MBIST_100_20_5	2	5,784	0.34	2.60	1.37	0.96
MBIST_100_100_5	3	6,246	0.35	2.56	1.14	0.98

Table B.11: A Hybrid Protection Scheme: Overhead

Initial Design		**Overhead**			
(1) Benchmark	(2) $\#viol.$	(3)$\#it.$	(4) %$\triangle$ HW_{filter}	(5) t_{struct}[m:s]	(6) t_{filter}[m:s]
q12710	501	3	0.98	00:05	00:51
a586710	1.820	2	0.78	02:10	01:46
p34392	502	3	4.17	00:30	01:48
t512505	924	2	1.65	02:54	00:59
p22810	28,941	2	7.21	06:40	10:25
p93791	18,610	2	5.50	08:34	13:51
MBIST_2_20_20	8,519	1	10.05	00:58	01:36
MBIST_5_20_20	1,020	0	6.42	01:20	02:16
MBIST_5_100_20	2,559	1	23.79	14:33	10:23
MBIST_5_100_100	9,828	1	3.44	18:46	19:54
MBIST_20_20_20	3,779	1	3.41	14:01	09:27
MBIST_55_20_5	59,055	1	8.90	10:25	11:36
MBIST_100_20_5	22,084	1	10.54	05:48	07:23
MBIST_100_100_5	160,687	1	8.63	12:54	25:10

Table B.12: A Hybrid Protection Scheme Comparison: Filter Applicability Fraction

Initial Design		**Comparison**
(1) Benchmark	(2) # Violations	(3) %, filters only
q12710	501	84.0
a586710	1.820	67.3
p34392	502	79.2
t512505	924	94.6
p22810	28,941	93.1
p93791	18,610	94.2
MBIST_2_20_20	8,519	83.4
MBIST_5_20_20	1,020	100
MBIST_5_100_20	2,559	90.0
MBIST_5_100_100	9,828	99.2
MBIST_20_20_20	3,779	89.8
MBIST_55_20_5	59,055	85.1
MBIST_100_20_5	22,084	98.3
MBIST_100_100_5	160,687	99.8

Table B.13: A Hybrid Protection Scheme Comparison: Number of Structural Changes

Initial Design	**Comparison**	
(1) Benchmark	(2) #changes hybrid	(3) # changes struct [LAWW20]
q12710	38	63
a586710	47	186
p34392	48	71
t512505	36	174
p22810	499	1,922
p93791	463	1,891
MBIST_2_20_20	1,408	2,247
MBIST_5_20_20	0	125
MBIST_5_100_20	241	415
MBIST_5_100_100	78	1,020
MBIST_20_20_20	293	335
MBIST_55_20_5	6,857	7,658
MBIST_100_20_5	354	1,563
MBIST_100_100_5	195	10,376

Bibliography

[ABF90] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design, Revised Printing*. IEEE Press, NY, 1st edition, 1990. ISBN: 0-7803-1062-4.

[AKS$^+$18] A. Atteya, M. A. Kochte, M. Sauer, P. Raiola, B. Becker, and H.-J. Wunderlich. Online Prevention of Security Violations in Reconfigurable Scan Networks. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, May 2018.

[ALR04] A. Avižienis, J.-C. Laprie, and B. Randell. Dependability and Its Threats: A Taxonomy. In Renè Jacquart, editor, *Building the Information Society*, pages 91–120, Boston, MA, 2004. Springer US. ISBN: 978-1-4020-8157-6.

[APZM07] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra. Circuit Failure Prediction and Its Application to Transistor Aging. In *Proc. IEEE VLSI Test Symp.(VTS)*, pages 277–286, 2007. doi:10.1109/VTS.2007.22.

[BBK89] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Proc. Int'l Symp. on Circuits and Systems (ISCAS)*, pages 1929–1934 vol.3, May 1989.

[BFH$^+$21] T. Balyo, N. Froleyks, M. J. H. Heule, M. Iser, M. Järvisalo, and M. Suda. Solver and Benchmark Descriptions. In *Proc. of SAT COMPETITION*, 2021.

[BHW14] C. Braun, S. Halder, and H.-J. Wunderlich. A-ABFT: Autonomous Algorithm-Based Fault Tolerance for Matrix Multiplications on Graphics Processing Units. In *Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 443–454, 2014. doi:10.1109/DSN.2014.48.

[Big74] N. Biggs. *Algebraic Graph Theory*. Cambridge Mathematical Library. Cambridge University Press, 1974. ISBN: 9780521203357.

[BKW14] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich. Access Port Protection for Reconfigurable Scan Networks. *Journal of Electronic Testing: Theory and Applications (JETTA)*, 30(6):711–723, 2014.

[BKW15a] R. Baranowski, M. A. Kochte, and H. Wunderlich. Fine-grained access management in reconfigurable scan networks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 34(6):937–946, June 2015. doi:10.1109/TCAD.2015.2391266.

[BKW15b] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich. Reconfigurable Scan Networks: Modeling, Verification, and Optimal Pattern Generation. *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, 20(2):1 – 30, 2015.

[BKW20] S. Brandhofer, M. A. Kochte, and H. Wunderlich. Synthesis of Fault-Tolerant Reconfigurable Scan Networks. In *Proc. Design, Automation Test in Europe Conf. Exhibition (DATE)*, pages 798–803, Mar. 2020. doi:10.23919/DATE48585.2020.9116525.

[Boo12] *Boolean Reasoning: The Logic of Boolean Equations*. Dover Publications, 2012. ISBN: 978-0486427850.

[BV] S. Boyd and L. Vanderberghe. *Convex Optimization, year = 2004, publisher = Cambridge University Press, page = 129, isbn = 978-0-521-83378-3*.

[CDRS18] R. Cantoro, A. Damljanovic, M. S. Reorda, and G. Squillero. A New Technique to Generate Test Sequences for Reconfigurable Scan Networks. In *Proc. IEEE Int'l Test Conf. (ITC)*, pages 1–9, 2018. doi:10.1109/TEST.2018.8624742.

[CDRS20] R. Cantoro, A. Damljanovic, M. S. Reorda, and G. Squillero. A Novel Sequence Generation Approach to Diagnose Faults in Reconfigurable Scan Networks. *IEEE Trans. on Computers*, 69(1):87–98, 2020.

[CMR$^+$15] R. Cantoro, M. Montazeri, M. S. Reorda, F. G. Zadegan, and E. Larsson. On the Testability of IEEE 1687 Networks. In *Proc. Asian Test Symp. (ATS)*, pages 211–216, 2015. doi:10.1109/ATS.2015.7447934.

[CMR$^+$19] W. Cheng, G. Mrugalski, J. Rajski, M. Trawka, and J. Tyszer. On Cyclic Scan Integrity Tests for EDT-based Compression. In *Proc. VLSI Test Symp. (VTS)*, pages 1–6, 2019. doi:10.1109/VTS.2019.8758670.

[CR36] A. Church and J. B. Rosser. Some properties of conversion. 1(2):472–482., 1936. doi:10.2307/2268572.

[CSSS18] R. Cantoro, L. San Paolo, M. Sonza Reorda, and G. Squillero. An Evolutionary Technique for Reducing the Duration of Reconfigurable Scan Network Test. In *Proc. IEEE Int.-l Symp. on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, pages 129–134, Apr. 2018.

[CVTR20] A. L. Crouch, B. G. Van Treuren, and J. Rearick. P1687.1: Accessing Embedded 1687 Instruments using Alternate Device Interfaces other than JTAG. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, May 2020.

[CZP$^+$18] R. Cantoro, F. G. Zadegan, M. Palena, P. Pasini, E. Larsson, and M. S. Reorda. Test of Reconfigurable Modules in Scan Networks. *IEEE Trans. on Computers (TC)*, 67(12):1806–1817, 2018. doi:10.1109/TC.2018.2834915.

[DDFR13] J. Da Rolt, G. Di Natale, M. Flottes, and B. Rouzeyre. A smart test controller for scan chains in secure circuits. In *Proc. IEEE Int'l On-Line Testing Symp. (IOLTS)*, pages 228–229, Jul. 2013.

[DFDR19] M. Da Silva, M. Flottes, G. Di Natale, and B. Rouzeyre. Preventing scan attacks on secure circuits through scan chain encryption. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(3):538–550, 2019.

[DH90] S. Dutt and J. P. Hayes. On designing and reconfiguring k-fault-tolerant tree architectures. *IEEE Trans. on Computers*, 39(4):490–503, 1990. doi:10.1109/12.54842.

[DJP$^+$19] A. Damljanovic, A. Jutman, M. Portolan, E. Sanchez, G. Squillero, and A. Tsertov. Simulation-based Equivalence Checking between IEEE 1687 ICL and RTL. In *Proc. IEEE Int.-l Test Conf. (ITC)*, pages 1–8, 2019. doi:10.1109/ITC44170.2019.9000181.

[DJST19] A. Damljanovic, A. Jutman, G. Squillero, and A. Tsertov. Post-Silicon Validation of IEEE 1687 Reconfigurable Scan Networks. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, May 2019. doi:10.1109/ETS.2019.8791546.

[DLL62] M. Davis, G. Logemann, and D. Loveland. A Machine Program for Theorem-Proving. *Commun. ACM*, 5(7):394–397, Jul. 1962.

[dN11] Organización Internacional de Normalización. ISO 26262: Road Vehicles : Functional Safety. 2011.

[Dom12] D. M. Doman. *Engineering the CMOS Library: Enhancing Digital Design Kits for Competitive Silicon*. Wiley, 2012.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6(2):182–197, 2002.

[DSGJ19] A. Damljanovic, G. Squillero, C. C. Güursoy, and M. Jenihhin. On NBTI-induced Aging Analysis in IEEE 1687 Reconfigurable Scan Networks. In *Proc. IFIP/IEEE 27th Int.-l Conf. on Very Large Scale Integration (VLSI-SoC)*, pages 335–340, 2019. doi:10.1109/VLSI-SoC.2019.8920313.

[DT19] A. Das and N. A. Touba. A Graph Theory Approach towards IJTAG Security via Controlled Scan Chain Isolation. In *Proc. IEEE VLSI Test Symp. (VTS)*, pages 1–6, 2019.

[EKC18] R. Elnaggar, R. Karri, and K. Chakrabarty. Securing IJTAG against data-integrity attacks. In *Proc. IEEE VLSI Test Symp. (VTS)*, pages 1–6, Apr. 2018. doi:10.1109/VTS.2018.8368642.

[EKC21] R. Elnaggar, R. Karri, and K. Chakrabarty. Security Against Data-Sniffing and Alteration Attacks in IJTAG. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 40(7):1301–1314, 2021. doi:10.1109/TCAD.2020.3019167.

[EKD$^+$03] D. Ernst, Nam Sung Kim, S. Das, S. Pant, R. Rao, Toan Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pages 7–18, 2003. doi:10.1109/MICRO.2003.1253179.

[ES04] N. Eén and N. Sörensson. An Extensible SAT-solver. In E. Giunchiglia and A. Tacchella, editors, *Theory and Applications of Satisfiability Testing*, pages 502–518, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN: 978-3-540-24605-3.

[EZ17] Ch. Eychenne and Y. Zorian. An effective functional safety infrastructure for system-on-chips. In *Proc. IEEE Int.-l Symp. on On-Line Testing and Robust System Design (IOLTS)*, pages 63–66, 2017. doi:10.1109/IOLTS.2017.8046235.

[FF58] L. R. Ford and D. R. Fulkerson. A Suggested Computation for Maximal Multi-Commodity Network Flows. *Management Science*, 5:97–101, 1958. doi:10.1287/mnsc.1040.0269.

[GCDE17] S. Gupta, A. Crouch, J. Dworak, and D. Engels. Increasing IJTAG bandwidth and managing security through parallel locking-SIBs. In *Proc. IEEE Int'l Test Conf. (ITC)*, pages 1–10, Nov. 2017.

[Gre] L. Greenemeier. iPhone Hacks Annoy AT&T but Are Unlikely to Bruise Apple. Available `https://www.scientificamerican.com/article/iphone-hacks-annoy-at/`. Accessed December 09, 2021 [Online].

[Gur19] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2019. http://www.gurobi.com.

[GWD18] S. Gupta, J. Wu, and Jennifer Dworak. Efficient parallel testing: A configurable and scalable broadcast network design using IJTAG. In *Proc. IEEE VLSI Test Symp. (VTS)*, pages 1–6, 2018. doi:10.1109/VTS.2018.8368641.

[HHD20] P. Habiby, S. Huhn, and R. Drechsler. Power-aware Test Scheduling for IEEE 1687 Networks with Multiple Power Domains. In *Proc. IEEE Int.-l Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6, 2020. doi:10.1109/DFT50435.2020.9250874.

[HHD21] P. Habiby, S. Huhn, and R. Drechsler. Optimization-based Test Scheduling for IEEE 1687 Multi-Power Domain Networks Using Boolean Satisfiability. In *Proc. Int.-l Conf. on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–4, 2021. doi:10.1109/DTIS53253.2021.9505098.

[HHS07] A. Hall, S. Hippler, and M. Skutella. Multicommodity Flows Over Time: Efficient Algorithms and Complexity. *Theoretical Computer Science*, 379(3):387 – 404, 2007.

[HP13] Z. Hanna and V. M. Purri. Verifying Security Aspects of SoC Designs with Jasper App, Apr. 2013. https://www.edn.com/.

[HWZ+07] A.-W. Hakmi, H.-J. Wunderlich, C.G. Zoellin, A. Glowatz, F. Hapke, J. Schloeffel, and L. Souef. Programmable deterministic Built-In Self-Test. In *Proc. IEEE Int.-l Test Conf.*, pages 1–9, 2007. doi:10.1109/TEST.2007.4437611.

[IEE13] IEEE Standard for Test Access Port and Boundary-Scan Architecture. *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)*, pages 1–444, May 2013.

[IEE14] IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device. *IEEE Std 1687-2014*, pages 1–283, Dec. 2014.

[IK16] A. M. Y. Ibrahim and H. G. Kerkhoff. Analysis and design of an on-chip retargeting engine for IEEE 1687 networks. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, May 2016. doi:10.1109/ETS.2016.7519301.

[IK19] A. M. Y. Ibrahim and H. G. Kerkhoff. An On-chip IEEE 1687 Network Controller for Reliability and Functional Safety Management of System-on-Chips. In *Proc. Int'l. Test Conf. in Asia (ITC-Asia)*, pages 109–114, 2019. doi:https://doi.org/10.1109/ITC-Asia.2019.00032.

[JRBS06] V. Joshi, R.R. Rao, D. Blaauw, and D. Sylvester. Logic SER reduction through flip flop redesign. In *Int.-l Symp. on Quality Electronic Design (ISQED)*, pages 1–6, Mar. 2006. doi:10.1109/ISQED.2006.82.

[KA84] K.-H. Huang and J. A. Abraham. Algorithm-Based Fault Tolerance for Matrix Operations. *IEEE Trans. on Computers*, C-33(6):518–528, 1984. doi:10.1109/TC.1984.1676475.

[KBS+16] M. A. Kochte, R. Baranowski, M. Sauer, B. Becker, and H.-J. Wunderlich. Formal Verification of Secure Reconfigurable Scan Network Infrastructure. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, May 2016. doi:http://dx.doi.org/10.1109/ETS.2016.7519290.

[KBSW16] M. A. Kochte, R. Baranowski, M. Schaal, and H. Wunderlich. Test Strategies for Reconfigurable Scan Networks. In *Proc. Asian Test Symp. (ATS)*, pages 113–118, 2016. doi:10.1109/ATS.2016.35.

[KBW17] M. A. Kochte, R. Baranowski, and H.-J. Wunderlich. Trustworthy Reconfigurable Access to On-Chip Infrastructure. In *Proc. IEEE Int'l Test Conf. in Asia (ITC-Asia)*, Sep. 2017.

[KCK21] S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, pages 1 – 36, 2021.

[KD18] S. Kan and J. Dworak. IJTAG Integrity Checking with Chained Hashing. In *Proc. IEEE Int.-l Test Conf. (ITC)*, pages 1–10, 2018. doi:10.1109/TEST.2018.8624777.

[KDD16] S. Kan, J. Dworak, and J. G. Dunham. Echeloned IJTAG data protection. In *Proc. IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, pages 1–6, 2016. doi:10.1109/AsianHOST.2016.7835558.

[KEGT17] S. Kiamehr, M. Ebrahimi, M. S. Golanbari, and M. B. Tahoori. Temperature-Aware Dynamic Voltage Scaling to Improve Energy Efficiency of Near-Threshold Computing. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 25(7):2017–2026, 2017. doi:10.1109/TVLSI.2017.2669375.

[KG14] J. Keller and R. Gerhards. PEELSCHED: a Simple and Parallel Scheduling Algorithm for Static Taskgraphs. *PARS: Parallel-Algorithmen, -Rechnerstrukturen und -Systemsoftware*, 28, Oct. 2014. doi:10.1007/BF03341989.

[KK07] I. Koren and C. M. Krishna. *Fault-Tolerant Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2007. ISBN: 0120885255.

[KMM$^+$20] B. Kaczmarek, G. Mrugalski, N. Mukherjee, J. Rajski, Ł. Rybak, and J. Tyszer. Test Sequence-Optimized BIST for Automotive Applications. In *Proc. IEEE European Test Symposium (ETS)*, pages 1–6, 2020. doi:10.1109/ETS48528.2020.9131585.

[KMM$^+$21] B. Kaczmarek, G. Mrugalski, N. Mukherjee, A. Pogiel, J. Rajski, L. Rybak, and J. Tyszer. LBIST for Automotive ICs with Enhanced Test Generation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, pages 1–1, 2021. doi:10.1109/TCAD.2021.3100741.

[KSRG$^+$17] M. A. Kochte, M. Sauer, L. Rodríguez Gómez, P. Raiola, B. Becker, and H.-J. Wunderlich. Specification and Verification of Security in Reconfigurable Scan Networks. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, Limassol, Cyprus, May 2017. doi:10.1109/ETS.2017.7968247.

[KW90] A. Kunzmann and H.-J. Wunderlich. An Analytical Approach to the Partial Scan Problem. *Journal of Electronic Testing: Theory and Applications (JETTA)*, 1(2):163–174, 1990. doi:10.1007/BF00137392.

[KW16] M. A. Kochte and H.-J. Wunderlich. Dependable on-chip infrastructure for dependable MPSOCs. In *Proc. Latin-American Test Symposium (LATS)*, pages 183–188, 2016.

[KW18] M. A. Kochte and H.-J. Wunderlich. Self-Test and Diagnosis for Self-Aware Systems. *IEEE Design & Test*, 35(5):7–18, 2018. doi:http://dx.doi.org/10.1109/MDAT.2017.2762903.

[LA15] H. Liu and V. D. Agrawal. Securing IEEE 1687-2014 Standard Instrumentation Access by LFSR Key. In *Proc. IEEE Asian Test Symp. (ATS)*, pages 91–96, 2015.

[Lal00] Parag K. Lala. *Self-Checking and Fault-Tolerant Digital Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000. ISBN: 0124343708.

[LAR$^+$19] N. Lylina, A. Atteya, P. Raiola, M. Sauer, B. Becker, and H.-J. Wunderlich. Security Compliance Analysis of Reconfigurable Scan Networks. In *Proc. IEEE Int'l Test Conf. (ITC)*, pages 1–9, Nov. 2019.

[LAW21] N. Lylina, A. Atteya, and H.-J. Wunderlich. A Hybrid Protection Scheme for Reconfigurable Scan Networks. In *Proc. of the IEEE VLSI Test Symp. (VTS), Best Paper Award*, pages 1–7, Apr. 2021.

[LAWW20] N. Lylina, A. Atteya, C.-H. Wang, and H.-J. Wunderlich. Security Preserving Integration and Resynthesis of Reconfigurable Scan Networks. In *Proc. IEEE Int'l Test Conf. (ITC)*, pages 1–10, Nov. 2020.

[LB90] K.-J. Lee and M. A. Breuer. A Universal Test Sequence for CMOS Scan Registers. In *Proc. of the Custom Integrated Circuits Conference*, pages 28.5/1–28.5/4, 1990. doi:10.1109/CICC.1990.124822.

[LGRT11] M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich. Opt4J - A Modular Framework for Meta-heuristic Optimization. In *Proc. of the Genetic and Evolutionary Computing Conf. (GECCO)*, pages 1723–1730, Jul. 2011.

[LLY$^+$19] X. Li, W. Li, J. Ye, H. Li, and Y. Hu. Scan chain based attacks and countermeasures: A survey. *IEEE Access*, 7:85055–85065, 2019.

[LMZ21] E. Larsson, P. Murali, and Z. Zhang. Accessing general IEEE Std. 1687 networks via functional ports. In *Proc. IEEE Int.-l Test Conf. (ITC)*, pages 354–363, 2021. doi:10.1109/ITC50571.2021.00051.

[LSW20] C. Liu, E. Schneider, and H-J. Wunderlich. Using Programmable Delay Monitors for Wear-Out and Early Life Failure Prediction. In *Proc. ACM/IEEE Conference on Design, Automation Test in Europe (DATE'20)*, pages 1–6, 2020. doi:http://dx.doi.org/10.23919/DATE48585.2020.9116284.

[LTPP07] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic. Securing Designs against Scan-Based Side-Channel Attacks. *IEEE Transactions on Dependable and Secure Computing*, 4(4):325–336, Oct 2007. doi:10.1109/TDSC.2007.70215.

[LV62] R. E. Lyons and W. Vanderkulk. The Use of Triple-Modular Redundancy to Improve Computer Reliability. *IBM Journal of Research and Development*, 6(2):200–209, 1962. doi:10.1147/rd.62.0200.

[LWW21] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. Testability-Enhancing Resynthesis of Reconfigurable Scan Networks. In *Proc. of the IEEE Int.-l Test Conf.(ITC)*, pages 1–10, Virtual, Oct. 2021.

[LWW22a] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. A Complete Design-for-Test Scheme of Reconfigurable Scan Networks. *submitted*, pages 1–10, 2022.

[LWW22b] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. Online Periodic Test of Reconfigurable Scan Networks. In *To appear in Proc. of the IEEE Asian Test Symp.*, pages 1–6, Taichung, Taiwan, Nov. 2022.

[LWW22c] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. Robust Reconfigurable Scan Networks. In *Proc. Conf. on Design, Automation Test in Europe (DATE)*, pages 1–4, Mar. 2022.

[LWW22d] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. SCAR: Security Compliance Analysis and Resynthesis of Reconfigurable Scan Networks. *TCAD*, pages 1–14, 2022.

[LXM21] E. Larsson, Z. Xiang, and P. Murali. Graceful Degradation of Reconfigurable Scan Networks. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems (TVLSI)*, 29(7):1475–1479, 2021. doi:10.1109/TVLSI.2021.3076593.

[Mac99] D.J.C. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Trans. on Information Theory*, 45(2):399–431, 1999. doi:10.1109/18.748992.

[MCXBA99] A. Mili, B. Cukic, T. Xia, and R. Ben Ayed. Combining fault avoidance, fault removal and fault tolerance: an integrated model. In *14th IEEE International Conference on Automated Software Engineering*, pages 137–146, 1999. doi:10.1109/ASE.1999.802168.

[MIC02] E. J. Marinissen, V. Iyengar, and K. Chakrabarty. A Set of Benchmarks for Modular Testing of SOCs. In *Proc. IEEE Int'l. Test Conf. (ITC)*, pages 519–528, 2002. doi:10.1109/TEST.2002.1041802.

[Mil63] W. H. Mills. Some complete cycles on the n-cube. *Proceedings of the American Mathematical Society*, 14(4):640–643, 1963.

[MM97] S. R. Maka and E. J. McCluskey. ATPG for Scan Chain Latches and Flip-Flops. In *Proc. VLSI Test Symp. (VTS)*, pages 364–369, Apr. 1997. doi:10.1109/VTEST.1997.600306.

[MM00] S. Mitra and E. J. McCluskey. Which concurrent error detection scheme to choose? In *Proc. Int'l. Test Conf. (ITC)*, pages 985–994, 2000. doi:10.1109/TEST.2000.894311.

[Moo65] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, Apr. 1965. doi:10.1109/N-SSC.2006.4785860.

[Moo75] G. E. Moore. Progress In Digital Integrated Electronics. In *Proc. IEEE International Electron Devices Meeting*, volume 21, pages 11–13, Sep. 1975. doi:10.1109/N-SSC.2006.4804410.

[MR90] F. Maamari and J. Rajski. A method of fault simulation based on stem regions. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 9(2):212–220, 1990. doi:10.1109/43.46788.

[MSN10] S. Mitra, S. A. Seshia, and N. Nicolici. Post-Silicon Validation Opportunities, Challenges and Recent Advances. In *Proc. Design Automation Conf. (DAC)*, pages 12–17, 2010. doi:10.1145/1837274.1837280.

[MSVK06] S. Mahapatra, D. Saha, D. Varghese, and P.B. Kumar. On the generation and recovery of interface traps in MOSFETs subjected to NBTI, FN, and HCI stress. *IEEE Transactions on Electron Devices*, 53(7):1583–1592, 2006. doi:10.1109/TED.2006.876041.

[MZS+07] S. Mitra, M. Zhang, N. Seifert, TM Mak, and K. S. Kim. Built-In Soft Error Resilience for Robust System Design. In *Proc. IEEE Int.-l Conf. on Integrated Circuit Design and Technology (ICICDT)*, pages 1–6, May 2007. doi:10.1109/ICICDT.2007.4299587.

[NSV+17] A. Nahiyan, M. Sadi, R. Vittal, G. Contreras, D. Forte, and M. Tehranipoor. Hardware trojan detection through information flow security verification. In *Proc. IEEE Int'l Test Conf. (ITC)*, pages 1–10, Oct. 2017. doi:10.1109/TEST.2017.8242062.

[NTKW98] K. Nakamura, K. Takagi, S. Kimura, and K. Watanabe. Waiting false path analysis of sequential logic circuits for performance optimization. In *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design. Digest of Technical Papers (ICCAD)*, pages 392–395, Nov. 1998. doi:10.1145/288548.289059.

[ORM07] M. Omana, D. Rossi, and C. Metra. Latch Susceptibility to Transient Faults and New Hardening Approach. *IEEE Trans. on Computers*, 56(9):1255–1268, 2007. doi:10.1109/TC.2007.1070.

[Par13] T Parr. *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf, 2nd edition, 2013. ISBN: 1934356999, 9781934356999.

[PBH+11] I. Polian, B. Becker, S. Hellebrand, H.-J. Wunderlich, and P. Maxwell. Towards Variation-Aware Test Methods. In *Proc. IEEE European Test Symp.*, pages 219–225, 2011. doi:10.1109/ETS.2011.51.

[PH11] I. Polian and J. P. Hayes. Selective Hardening: Toward Cost-Effective Error Tolerance. *IEEE Design and Test of Computers*, 28(3):54–63, 2011. doi:10.1109/MDT.2010.120.

[Pim17] A. D. Pimentel. Exploring Exploration: A Tutorial Introduction to Embedded Systems Design Space Exploration. *IEEE Design and Test*, 34(1):77–90, 2017.

[PK00] R.K. Prasad and I. Koren. The effect of placement on yield for standard cell designs. In *Proceedings IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 3–11, 2000. doi:10.1109/DFTVS.2000.886968.

[Por16] M. Portolan. Accessing 1687 systems using arbitrary protocols. In *Proc. IEEE Int.-l Test Conf. (ITC)*, pages 1–9, 2016. doi:10.1109/TEST.2016.7805839.

[PRML20] M. Portolan, V. Reynaud, P. Maistri, and R. Leveugle. Dynamic Authentication-Based Secure Access to Test Infrastructure. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, 2020. doi:10.1109/ETS48528.2020.9131571.

[PV79] F. P. Preparata and J. Vuillemin. The cube-connected-cycles: A versatile network for parallel computation. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 140–147, 1979. doi:10.1109/SFCS.1979.43.

[QM06] Z. Quming and K. Mohanram. Gate sizing to radiation harden combinational logic. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 25(1):155–166, 2006. doi:10.1109/TCAD.2005.853696.

[RBT18] X. Ren, R. D. S. Blanton, and V. G. Tavares. Detection of IJTAG attacks using LDPC-based feature reduction and machine learning. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, May 2018. doi:10.1109/ETS.2018.8400684.

[RDN+14] J. Da Rolt, A. Das, G. Di Natale, M. Flottes, B. Rouzeyre, and I. Verbauwhede. Test Versus Security: Past and Present. *IEEE Trans. on Emerging Topics in Computing*, 2(1):50–62, Mar. 2014. doi:10.1109/TETC.2014.2304492.

[REP+05] J. Rearick, B. Eklow, K. Posse, A. Crouch, and B. Bennetts. IJTAG (internal JTAG): a step toward a DFT standard. In *Proc. IEEE Int'l Test Conf. (ITC)*, pages 1 – 8, 2005.

[RKA+18] P. Raiola, M. A. Kochte, A. Atteya, L. R. Gomez, H.-J. Wunderlich, B. Becker, and M. Sauer. Detecting and Resolving Security Violations in Reconfigurable Scan Networks. In *Proc. IEEE Int'l Symp. on On-Line Testing And Robust System Design (IOLTS)*, pages 91–96, Jul. 2018. doi:10.1109/IOLTS.2018.8474188.

[RPB20] P. Raiola, T. Paxian, and B. Becker. Minimal Witnesses for Security Weaknesses in Reconfigurable Scan Networks. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, May 2020.

[RTB+19] P. Raiola, B. Thiemann, J. Burchard, A. Atteya, N. Lylina, H.-J. Wunderlich, B. Becker, and M. Sauer. On Secure Data Flow in Reconfigurable Scan Networks. In *Proc. Conf. on Design, Automation Test in Europe (DATE)*, pages 1–6, Mar. 2019.

[RTKM04] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee. Embedded Deterministic Test. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(5):776–792, 2004. doi:10.1109/TCAD.2004.826558.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., USA, 1986. ISBN: 0471908541.

[SDJ16] K. Shibin, S. Devadze, and A. Jutman. On-line fault classification and handling in IEEE1687 based fault management system for complex SoCs. In *Proc. Latin-American Test Symp. (LATS)*, pages 69–74, 2016. doi:10.1109/LATW.2016.7483342.

[SKKN20] S. Sadeghi-Kohan, M. Kamal, and Z. Navabi. Self-Adjusting Monitor for Measuring Aging Rate and Advancement. *IEEE Transactions on Emerging Topics in Computing*, 8(3):627–641, 2020. doi:10.1109/TETC.2017.2771441.

[SRS+16] M. Soeken, P. Raiola, B. Sterin, B. Becker, G. De Micheli, and M. Sauer. *Proc. 12th Int'l Haifa Verification Conference (HVC)*, chapter SAT-Based Combinational and Sequential Dependency Computation, pages 1–17. Springer, 2016. ISBN: 978-3-319-49052-6.

[Tei12] J. Teich. Hardware/Software Codesign: The Past, the Present, and Predicting the Future. *Proc. of the IEEE*, 100(Special Centennial Issue):1411–1430, 2012.

[TFR$^+$19] B. Thiemann, L. Feiten, P. Raiola, B. Becker, and M. Sauer. On Integrating Lightweight Encryption in Reconfigurable Scan Networks. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–6, May 2019. doi:10.1109/ETS.2019.8791543.

[TJD$^+$16] A. Tsertov, A. Jutman, S. Devadze, M. S. Reorda, E. Larsson, F. G. Zadegan, R. Cantoro, M. Montazeri, and R. Krenz-Baath. A suite of IEEE 1687 benchmark networks. In *Proc. IEEE Int'l Test Conf. (ITC)*, pages 1–10, Nov. 2016. doi:10.1109/TEST.2016.7805840.

[TJSD18] A. Tsertov, A. Jutman, K. Shibin, and S. Devadze. IEEE 1687 Compliant Ecosystem for Embedded Instrumentation Access and In-Field Health Monitoring. In *Proc. IEEE AUTOTESTCON*, pages 1–9, 2018. doi:10.1109/AUTEST.2018.8532559.

[TKD$^+$07] J. Tschanz, N. S. Kim, S. Dighe, J. Howard, G. Ruhl, S. Vangal, S. Narendra, Y. Hoskote, H. Wilson, C. Lam, M. Shuman, C. Tokunaga, D. Somasekhar, S. Tang, D. Finan, T. Karnik, N. Borkar, N. Kurd, and V. De. Adaptive Frequency and Biasing Techniques for Tolerance to Dynamic Temperature-Voltage Variations and Aging. In *Proc. Int.-l Solid-State Circuits Conf.*, pages 292–604, Feb. 2007. doi:10.1109/ISSCC.2007.373409.

[TW11] M. Tehranipoor and C. Wang. *Introduction to Hardware Security and Trust*. Springer, 2011.

[UKW17] D. Ull, M. Kochte, and H. Wunderlich. Structure-Oriented Test of Reconfigurable Scan Networks. In *Proc. Asian Test Symp. (ATS)*, pages 127–132, 2017. doi:10.1109/ATS.2017.34.

[VBG$^+$08] J. Vial, A. Bosio, P. Girard, C. Landrault, S. Pravossoudovitch, and A. Virazel. Using TMR Architectures for Yield Improvement. In *IEEE Int.-l Symp. on Defect and Fault Tolerance of VLSI Systems (DFT)*, pages 7–15, Oct. 2008. doi:10.1109/DFT.2008.23.

[VGY$^+$16] E. Vattapparamban, İ. Güvenç, A. İ. Yurekli, K. Akkaya, and S. Uluağaç. Drones for smart cities: Issues in cybersecurity, privacy, and public safety. In *Proc. Int.-l Wireless Communications and Mobile Computing Conf. (IWCMC)*, pages 216–221, 2016. doi:10.1109/IWCMC.2016.7577060.

[vSVTR$^+$21] H.-M. von Staudt, B. Van Treuren, J. Rearick, M. Portolan, and M. Keim. Exploring and Comparing IEEE P1687.1 and IEEE 1687 Modeling of Non-TAP Interfaces. In *Proc. IEEE European Test Symp. (ETS)*, pages 1–10, 2021. doi:10.1109/ETS50041.2021.9465438.

[VTL79] J. Valdes, R. E. Tarjan, and E. L. Lawler. The Recognition of Series Parallel Digraphs. In *Proc. of the Annual ACM Symp. on Theory of Computing*, page 1–12, 1979. ISBN: 9781450374385.

[War62] S. Warshall. A Theorem on Boolean Matrices. *Journal of the ACM (JACM)*, 9(1):11–12, Jan. 1962. doi:10.1145/321105.321107.

[WLA+21] C.-H. Wang, N. Lylina, A. Atteya, T.-Y. Hsieh, and H.-J. Wunderlich. Concurrent Test of Reconfigurable Scan Networks for Self-Aware Systems. In *Proc. IEEE Int.-l Symp. on On-Line Testing And Robust System Design (IOLTS)*, pages 1–7, Virtual, Jun. 2021.

[WWW06] L.-T. Wang, C.-W. Wu, and X. Wen. *VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006. ISBN: 0123705975.

[XBO] Free60 SMC Hack. Available `https://free60project.github.io/wiki/SMC_Hack/`. Accessed December 09, 2021 [Online].

[YCD+08] F. Yang, S. Chakravarty, N. Devta-Prasanna, S. M. Reddy, and I. Pomeranz. Detection of Internal Stuck-open Faults in Scan Chains. In *Proc. Int'l. Test Conf. (ITC)*, pages 1–10, Dec. 2008. doi:10.1109/TEST.2008.4700577.

[YWK06] B. Yang, K. Wu, and R. Karri. Secure Scan: A Design-for-Test Architecture for Crypto Chips. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 25(10):2287–2293, Oct. 2006. doi:10.1109/TCAD.2005.862745.

[ZC20] Z. Zhong and K. Chakrabarty. IJTAG-Based Fault Recovery and Robust Microelectrode-Cell Design for MEDA Biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4921–4934, 2020. doi:10.1109/TCAD.2020.2986741.

[ZDCP14] A. Zygmontowicz, J. Dworak, A. Crouch, and J. Potter. Making it harder to unlock an LSIB: Honeytraps and misdirection in a P1687 network. In *Proc. Design, Automation Test in Europe Conf. (DATE)*, pages 1–6, 2014. doi:10.7873/DATE.2014.208.

[ZLJ+14] F. G. Zadegan, E. Larsson, A. Jutman, S. Devadze, and R. Krenz-Baath. Design, Verification, and Application of IEEE 1687. In *Proc. IEEE Asian Test Symp. (ATS)*, pages 93–100, Nov. 2014. doi:10.1109/ATS.2014.28.

[ZLT01] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm For Multiobjective Optimization. *Technical Report 103*, May 2001.

[ZLYC20] Z. Zhong, G. Li, Q. Yang, and K. Chakrabarty. Access-Time Minimization for the IJTAG Network Using Data Broadcast and Hardware Parallelism. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, pages 185 – 198, Apr. 2020.

[ZNL16] F. G. Zadegan, D. Nikolov, and E. Larsson. A self-reconfiguring IEEE 1687 network for fault monitoring. In *Proc. European Test Symp. (ETS)*, pages 1–6, 2016. doi:10.1109/ETS.2016.7519288.

[ZWPB08] C. G. Zoellin, H.-J. Wunderlich, I. Polian, and B. Becker. Selective Hardening in Early Design Steps. In *Proc. IEEE European Test Symp. (ETS)*, pages 185–190, May 2008.

Index

Boolean satisfiability, 113

dependability, 1, 146
  enhancement, 4
    fault-avoidance, 5, 57
    fault-forecasting, 5
    fault-removal, 5, 34
    fault-tolerance, 4, 34
  requirements, 1
    availability, 2
    confidentiality, 3
    functional safety, 3
    integrity, 3
    maintainability, 2
    reliability, 2

graph, 27
  acyclic graph, 27
  cyclic graph, 27
  directed graph, 27
  notation, 27
    infix, 27
    postfix, 28
    prefix, 27
    reverse Polish notation, 28
  tree, 27

optimization, 47
  Boolean optimization, 47
  Integer Linear Programming, 119

Reconfigurable Scan Network, 8, 20
  control primitive, 21
    scan multiplexer, 21
    segment insertion bit, 21
  control signal, 21
  dependability management, 8, 11, 33, 146
    applications, 9, 33
    integration, 146
    requirements, 9
  dependability property, 45
    robustness, 10, 34, 45, 149
    security compliance, 10, 38, 108, 148, 150
    testability, 10, 35, 70, 149, 150
  fault, 24, 54, 72
    control line, 25, 36, 55, 78
    detectability, 72
    interface, 24, 36, 54, 75
    scan segment, 26, 37, 55, 85
  graph model, 28
    child, 30
    domination, 30
    neighbor, 30

    - parent, 30
    - predecessors, 29
    - reachability, 29
    - reconvergence vertex, 30
    - reconvergency region, 30
    - series-parallel property, 52, 61, 97
    - successors, 29
  - operation, 22
    - CSU-operation, 23
    - scan configuration, 22
    - transition relation, 23
  - optimization criteria, 45
    - accessibility, 72
    - hardware overhead, 50
  - path, 29
    - active scan path, 22, 29
    - capture path, 23
    - functional path, 29
    - shift path, 23
    - structural path, 29
    - update path, 23
  - robustness, 34, 45, 149
    - criticality analysis, 51
    - selective hardening, 57
  - scan primitive, 20
    - control primitive, 21
    - scan segment, 20
  - scan segment, 20
    - control scan segment, 20
    - data scan segment, 20
  - security compliance, 10, 38, 108, 148
    - analysis, 38, 108, 148
    - integration, 39, 117
    - multi-user protection scheme, 128, 150
    - specification, 109
    - structural resynthesis, 117, 148
    - violation, 110, 116, 128
  - specification, 48
    - criticality, 48
    - security, 109
  - test, 35, 150
    - control line, 36, 78
    - design-for-test scheme, 71
    - interface, 36, 75
    - periodic test, 90
    - scan segment, 37, 85
    - test integration, 88
    - test scheduling, 93
  - testability, 35, 70, 149, 150
    - analysis, 78
    - control line, 78
    - enhancement, 75, 84, 85
    - interface, 75
    - scan segment, 85

- security, 3, 108
- silicon lifecycle management, 5
  - design stage, 6
  - field return phase, 7
  - in-field operation, 7
  - manufacturing test, 7
  - post-silicon validation, 6

Curriculum Vitae of the Author

CONTACT

lylina.natalia.1993@gmail.com

natalia-lylina-799566b0

ResearchGate: Natalia Lylina

+49 172 691 97 61

Maiden name: Kaptsova

TESTING SKILLS

Design-for-Test	3+ yrs
iJTAG Networks	3+ yrs
Scan Design & Test	3+ yrs
Test & Testability Enhancement	3+ yrs
Silicon Lifecycle (Post-Silicon, Volume Production, In-Field)	3+ yrs
System Dependability	3+ yrs

ALGORITHMIC SKILLS

Optimization (Integer Linear Programming, Pseudo-Boolean Optimization, Multi-Objective Optimization)	5+ yrs
Formal Verification (SAT, maxSAT)	3+ yrs
Machine Learning	3+ yrs

NATALIA LYLINA

Ph.D. Student

EDUCATION

Ph. D. - Natural Sciences, Informatics **2017 - ongoing**
University of Stuttgart, Institute of Computer Architecture and Computer Engineering (ITI)

Topic: Dependable Reconfigurable Scan Networks (*in english*)

M. Sc. - Computer Science & Engineering **2015 - 2017**
Moscow Power Engineering Institute (National Research University) - Moscow (Russia)

Grade: A (russian 5.0); **Master thesis:** Design and investigation of hardware and software components of the system on the chip (*in russian*).

M. Sc. - Engineering Informatics (Double Degree Program) **2014 - 2015**
Technical University of Ilmenau - Ilmenau (Germany)

Grade: A (german 1.0); **Master thesis:** Methodical developement of an optimizing assembler code compiler based on an object-oriented programming language (*in german*).

B. Sc. - Computer Science & Engineering **2010- 2014**
Moscow Power Engineering Institute (National Research University) - Moscow (Russia)

Grade: A (russian 5.0); **Bachelor thesis:** Development of a microprocessor control system prototype (*in russian*).

WORK EXPERIENCE

Research Assistant in the group of Prof. Hans-Joachim Wunderlich **2017 - ongoing**
University of Stuttgart, ITI, Stuttgart (Germany)

Research: Reconfigurable Scan Networks (RSNs, also iJTAG networks, IEEE Std. 1687) flexibly access the dependability instruments, such as monitors, sensors, Built-In-Self-Test registers. My research covers:

- *Various dependability aspects of RSNs*: testability, robustness, and security compliance with the underlying design;
- *Various lifecycle phases of the system*: starting from the RSN design and resynthesis, through post-silicon validation and volume test to the further in-field operation.
- *Different techniques*: Scalable modeling enables further precise analysis of specific dependability properties and efficient dependability-enhancing resynthesis.

Result: An automated framework for designing dependable RSNs.

Teaching, reporting and organization:

PROGRAMMING SKILLS

Java/C++	5+ yrs
Python	3+ yrs
VHDL	5+ yrs

OTHER SKILLS

Teaching	3+ yrs
LaTeX, Microsoft, SAP	3+ yrs
Linux, Ubuntu	5+ yrs

LANGUAGES

Russian	Native speaker
English	Professional
German	Professional

- Actively contributed to project proposals (ACCESS-2) and reports (SHIVA);
- Actively involved in organizing the SHIVA workshop at the European Test Symposium 2019;
- Organized seminars on modern research topics for master students (Scan Test; Machine Learning for Test and Diagnosis; etc.) and group-internal research seminars.

Junior Manager (Part-time job) **2011 - 2017**
Avicon Technologies, Moscow (Russia)

I have been working as a part-time junior manager and have been responsible for preparing the documents and some accounting tasks.

PROJECTS

ACCESS-2 (DFG) **2019-ongoing**
Languages: C++; Java, Python, VHDL;
Tools: CLion, Intellij idea, PyCharm, Modelsim

This project targets the in-field operation of RSNs for accessing the embedded instruments. I am responsible for developing algorithms for analyzing and enhancing the testability of RSNs and ensuring their reliable operation, as well as developing online test methods for RSNs.

Intelligent Methods for Test and Reliability (Graduate School, Advantest) **2019-ongoing**

I am involved in an interdisciplinary project which aims to utilize intelligent methods, such as machine learning and artificial intelligence, to enhance the reliability of modern devices.

SHIVA (BW-Stiftung) **2017-2019**
Languages: C++; Assembly, VHDL;
Tools: CLion, Intellij idea, Design Compiler

The focus of the SHIVA project is on the security aspects of RSN integration. Security violations might arise if an RSN is integrated improperly. I have developed an efficient method for precisely identifying the violations and resolving them efficiently by the presented security-enhancing resynthesis method.

ViSARD **2014-2017**
Languages: C++; Assembly, VHDL;
Tools: Xilinx Vivado, Modelsim

I have been responsible for enhancing the design of an arithmetical logic unit (ALU) block of a floating-point softcore processor and for developing an optimizing compiler for this processor. The parallelism within an assembly code is exploited in the presented compiler. The developed technique significantly reduced the overall runtime of the target programs and satisfied the timing requirements and hardware constraints.

AWARDS

PhD Forum Award

Design, Automation and Test in Europe Conference (DATE'22)
PhD thesis title: Dependable Reconfigurable Scan Networks

Best Paper Award

IEEE VLSI Test Symposium (VTS'21)
Paper title: "A Hybrid Protection Scheme for Reconfigurable Scan Networks".

DAAD Scholarship

German Academic Exchange Service scholarship (2014-2015)

PUBLICATIONS

- N. Lylina, C.-H. Wang, H.-J. Wunderlich **"Online Periodic Test of Reconfigurable Scan Networks"**, to appear in Proc. of ATS'22
- N. Lylina, C.-H. Wang, H.-J. Wunderlich **"Robust Reconfigurable Scan Networks"**, in Proc. of DATE'22
- N. Lylina, C.-H. Wang, H.-J. Wunderlich **"Using Reconfigurable Scan Networks to Support Silicon Lifecycle Management"**, in Proc. European SLM Workshop at DATE'22
- H. Amrouch, J. Anders, S. Becker, M. Betka, G. Bleher, P. Domanski, N. Elhamawy, T. Ertl, A. Gatzastras, P. Genssler, S. Hasler, M. Heinrich, A. van Hoorn, H. Jafarzadeh, I. Kallfass, F. Klemme, S. Koch, R. Küsters, A. Lalama, R. Latty, Y. Liao, N. Lylina, Z. P. Najafi-Haghi, D. Pflüger, I. Polian, J. Rivoir, M. Sauer, D. Schwachhofer, S. Templin, C. Volmer, S. Wagner, D. Weiskopf, H.-J. Wunderlich, B. Yang, M. Zimmermann **"Intelligent Methods for Test and Reliability"**, in Proc. of DATE'22
- N. Lylina, C.-H. Wang, H.-J. Wunderlich **"SCAR: Security Compliance Analysis and Resynthesis of Reconfigurable Scan Networks"**, in TCAD'22
- N. Lylina, C.-H. Wang, H.-J. Wunderlich **"Testability Enhancing Resynthesis of Reconfigurable Scan Networks"**, in Proc. of ITC'21
- C.-H. Wang, N. Lylina, A. Atteya, T.-Y. Hsieh and H.-J. Wunderlich **"Concurrent Test of Reconfigurable Scan Networks for Self- Aware Systems"**, in Proc. of IOLTS'21
- N. Lylina, A. Atteya, H.-J. Wunderlich **"A Hybrid Protection Scheme for Reconfigurable Scan Networks"**, in Proc. of VTS'21
- N. Lylina, A. Atteya, C.-H. Wang, H.-J. Wunderlich **"Security Preserving Integration and Resynthesis of Reconfigurable Scan Networks"**, in Proc. of ITC'20
- N. Lylina, A. Atteya, P. Raiola, M. Sauer, B. Becker, H.-J. Wunderlich **"Security Compliance Analysis of Reconfigurable Scan Networks"**, in Proc. of ITC'19
- P. Raiola, B. Thiemann, J. Burchard, A. Atteya, N.Lylina, H.-J. Wunderlich, B. Becker, M. Sauer **"On Secure Data Flow in Reconfigurable Scan Networks"**, in Proc. of DATE'19

Publications of the Author

In the following, all former publications of the author are listed in chronological order.

Journal Publications

[LWW22d] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. SCAR: Security Compliance Analysis and Resynthesis of Reconfigurable Scan Networks. *TCAD*, pages 1–14, 2022.

[LWW22a] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. A Complete Design-for-Test Scheme of Reconfigurable Scan Networks. *JETTA*, pages 1–19, 2022.

Conference Proceedings

[RTB+19] P. Raiola, B. Thiemann, J. Burchard, A. Atteya, N. Lylina, H.-J. Wunderlich, B. Becker, and M. Sauer. On Secure Data Flow in Reconfigurable Scan Networks. In *Proc. Conf. on Design, Automation Test in Europe (DATE)*, pages 1–6, Mar. 2019.

[LAR+19] N. Lylina, A. Atteya, P. Raiola, M. Sauer, B. Becker, and H.-J. Wunderlich. Security Compliance Analysis of Reconfigurable Scan Networks. In *Proc. IEEE Int'l Test Conf. (ITC)*, pages 1–9, Nov. 2019.

[LAWW20] N. Lylina, A. Atteya, C.-H. Wang, and H.-J. Wunderlich. Security Preserving Integration and Resynthesis of Reconfigurable Scan Networks. In *Proc. IEEE Int'l Test Conf. (ITC)*, pages 1–10, Nov. 2020.

[LAW21] N. Lylina, A. Atteya, and H.-J. Wunderlich. A Hybrid Protection Scheme for Reconfigurable Scan Networks. In *Proc. of the IEEE VLSI Test Symp. (VTS), Best Paper Award*, pages 1–7, Apr. 2021.

[WLA+21] C.-H. Wang, N. Lylina, A. Atteya, T.-Y. Hsieh, and H.-J. Wunderlich. Concurrent Test of Reconfigurable Scan Networks for Self-Aware Systems. In *Proc. IEEE Int.-l Symp. on On-Line Testing And Robust System Design (IOLTS)*, pages 1–7, Virtual, Jun. 2021.

[LWW21] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. Testability-Enhancing Resynthesis of Reconfigurable Scan Networks. In *Proc. of the IEEE Int.-l Test Conf.(ITC)*, pages 1–10, Virtual, Oct. 2021.

[LWW22c] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. Robust Reconfigurable Scan Networks. In *Proc. Conf. on Design, Automation Test in Europe (DATE)*, pages 1–4, Mar. 2022.

[AAB$^+$22] H. Amrouch, J. Anders, S. Becker, M. Betka, G. Bleher, P. Domanski, N. Elhamawy, T. Ertl, A. Gatzastras, P. R. Genssler, S. Hasler, M. Heinrich, A. van Hoorn, H. Jafarzadeh, I. Kallfass, F. Klemme, S. Koch, R. Küsters, A. Lalama, R. Latty, Y. Liao, N. Lylina, Z. P. Najafi-Haghi, D. Pflüger, I. Polian, J. Rivoir, M. Sauer, D. Schwachhofer, S. Templin, C. Volmer, S. Wagner, D. Weiskopf, H.-J. Wunderlich, B. Yang, and M. Zimmermann. Intelligent Methods for Test and Reliability. In *Proc. Conf. on Design, Automation Test in Europe (DATE)*, page 1–6, 03 2022.

[LWW22b] N. Lylina, C.-H. Wang, and H.-J. Wunderlich. Online Periodic Test of Reconfigurable Scan Networks. In *To appear in Proc. of the IEEE Asian Test Symp.*, pages 1–6, Taichung, Taiwan, Nov. 2022.

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.

Natalia Lylina