Journal of Electronic Testing (2022) 38:603–621
https://doi.org/10.1007/s10836-022-06038-3

A Complete Design-for-Test Scheme for Reconfigurable Scan Networks

Natalia Lylina[1] · Chih-Hao Wang[1] · Hans-Joachim Wunderlich[1]

Received: 7 July 2022 / Accepted: 19 November 2022 / Published online: 19 January 2023

Abstract
Reconfigurable Scan Networks (RSNs) are widely used for accessing instruments offline during debug, test and validation, as well as for performing system-level-test and online system health monitoring. The correct operation of RSNs is essential, and RSNs have to be thoroughly tested. However, due to their inherently sequential structure and complex control dependencies, large parts of RSNs have limited observability and controllability. As a result, certain faults at the interfaces to the instruments, control primitives and scan segments remain undetected by existing test methods. In the paper at hand, Design-for-test (DfT) schemes are developed to overcome the testability problems e.g. by resynthesizing the initial design. A DfT scheme for RSNs is presented, which allows detecting all single stuck-at-faults in RSNs by using existing test generation techniques. The developed scheme analyzes and ensures the testability of all parts of RSNs, which include scan segments, control primitives, and interfaces to the instruments. Therefore, the developed scheme is referred to as a *complete DfT scheme*. It allows for a test integration to cover multiple fault locations can with a single efficient test sequence and to reduce overall test cost.

Keywords Reconfigurable scan networks · Design-for-Test · Test · Debug · Diagnosis

1 Introduction

Reconfigurable Scan Networks (RSNs) offer flexible access to embedded instruments via scan segments thro-ughout the system lifecycle. They are standardized by IEEE Std. 1149.1-2013 [14] and IEEE Std. 1687-2014 [15]. To ensure the correct operation of RSNs and their interaction with the instruments, RSNs themselves must be thoroughly tested. Due to the low observability and controllability of certain parts of RSNs, some faults may be undetectable by the existing methods. In this paper, we present a scheme to enhance the design of RSNs in a way that all single stuck-at-faults in RSNs are detectable by using existing Automated Test Pattern Generation (ATPG) tools.

Initially, more focus has been put on using RSNs offline, e.g. for post-silicon validation (PSV) and manufacturing test and diagnosis. Recent standardization efforts (IEEE P2654 standard proposal [16], also discussed in [26]) suggest using RSNs to access registers needed for the system-level test. Research papers [13, 19, 29] use RSNs to perform health monitoring and dependability management. All these applications rely on the correct operation of RSNs, which may become a system dependability bottleneck. Already a single fault in an RSN may corrupt scan paths, erroneous data may be fetched by RSNs, and instruments may become inaccessible. During post-silicon validation, it may prevent extracting the complete validation data from a device. Runtime-critical instruments such as Adaptive Voltage and Frequency Scaling, temperature control, etc., may become inaccessible due to a fault in an RSN. Erroneous fault-handling mechanisms may be triggered by faulty RSNs. As a result, even a system failure may occur.

Faults in conventional scan chains can be tested by using flush test sequences shifted through a scan chain [22]. These sequences ensure the integrity of the scan cells and their interconnection. Typically used flush sequences 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 shifted into the 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.

Responsible editor: K.-J. Lee

✉ Natalia Lylina
natalia.lylina@informatik.uni-stuttgart.de

[1] Institute of Computer Architecture and Computer Engineering, University of Stuttgart, Stuttgart, Germany

For conventional scan chains, flush sequences test faults for certain fault models, such as stuck-at faults, delay faults, or consider broken scan chains. To detect the location of a fault within a scan chain, the responses from the applied Automated Test Pattern Generator (ATPG) test vectors need to be analyzed. The paper at hand focuses on test and testability questions. Therefore, fault localization is out of the scope of this paper.

Compared to conventional scan chains, testing RSNs is even more challenging. It is due to their high sequential depth, the distributed control structure, as well as the complex sequential and combinational dependencies [1]. Faults in RSNs not only affect the shift logic of the scan chain segments but also reside at the interfaces to the instruments and the control primitives. Additionally, specific fault effects in RSNs are observable only for certain configurations, and sequential test pattern generation for such faults is unfeasible for large networks. The known test solutions, presented in [4, 6, 12, 18], e.g., may not detect all the faults due to low observability and controllability of some RSN components.

Design-for-test (DfT) methods can be applied to enhance the testability of specific parts of RSNs. The term "DfT for RSNs" describes changes of the RSN design which support a more efficient test generation with higher fault coverage, a more efficient test application and more efficient test patterns for the RSN. The preliminary papers [23, 32, 34] developed DfT approaches to ensure the testability of the three components of a Reconfigurable Scan Network:

- **Scan interfaces** in [32]: The testability of the update 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.
- **Control primitives** in [23]: The testability of control primitives is enhanced. Existing methods test the control primitives by observing the length of an activated path [1, 3, 6, 10, 18, 32, 34], 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 an 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.
- **Scan segments** in [34]: The test of the scan shift logic is enabled by integrating a compact Built-In Self-Test (BIST) structure. This BIST structure generates of a short presequence that tests the shift logic of the currently activated scan path.

The paper at hand combines and extends the preliminary results in [23, 32, 34] by presenting a Design-for-Test (DfT) scheme which addresses all parts of an RSN. After applying the proposed DfT techniques, state-of-the-art test generation algorithms for RSNs like [1, 4, 6, 18] are able to generate test sequences which detect all stuck-at faults and obtain complete stuck-at-fault coverage. Therefore, the developed scheme is referred to as a *complete* DfT scheme. The presented scheme requires negligible hardware overhead and supports test generation with respect to more complex fault models as well. An efficient test integration scheme is developed in a way that a single test sequence covers multiple fault locations in an enhanced RSN, and the overall test application time in terms of the overall test application time is significantly reduced.

The extensions and improvements are in detail:

1. A *complete DfT scheme* is presented for the first time. It considers all fault locations at the scan interfaces to the instruments, the scan segments and the control primitives. First, it identifies precisely untestable faults, and then it re-synthesizes the corresponding components with negligible hardware costs. In the resulting RSN structure, existing ATPG techniques can detect all stuck-at faults.
2. For the resulting RSNs, an efficient *test integration scheme* is developed which allows to reduce the overall test access time compared to applying the previously developed methods [23, 32, 34] individually. The generated test sequences are capable to cover multiple fault locations at a time. Each sequence contains a workload sequence and a short presequence. The workload sequence tests faults at the scan interfaces and control primitives. The self-generated presequence tests the shift logic of the segments on the activated path.
3. The complete scheme is *evaluated on a comprehensive set of benchmarks*. The experimental results show the effectiveness and scalability of the developed approach.

The remainder of this paper is organized as follows. The background information about RSNs and their modeling, as well as the considered fault models, are presented in Section 2. Section 3 summarizes the existing methods to test RSNs and highlights their limitations. Section 4 provides an overview of the developed DfT scheme. In Section 5, a DfT scheme is presented for update registers and interfaces to instruments. Section 6 presents a testability enhancement technique for control primitives. Section 7 provides the details about the scan segment test. Section 8 discusses the overall test integration procedure. In Section 9, experimental results are discussed, and Section 10 concludes the paper.

2 Background

This section presents background information about RSNs, the graph-based RSN model and the considered fault models.

2.1 Reconfigurable Scan Networks (RSNs)

In Fig. 1, the internal instruments (shown in orange), such as sensors and BIST registers, are accessed through RSNs for observation and control. The test data registers ($s1 \dots s8$) access the instruments through a parallel interface and are also referred to as the scan segments. The configuration segments ($cs0 \dots cs3$), shown in yellow, determine the values of control signals. Those signals control the state of *control primitives* such as scan multiplexers (shown with the circles with the matching indices in Fig. 1). Control primitives determine the currently configured non-circular scan path from a primary scan input to a primary scan output port of an RSN. Such a path is commonly referred to as an Active Scan Path (ASP) and is shown with a blue dashed line in Fig. 1. In the considered example, an initial ASP traverses the control registers $cs0$, $cs2$, and $cs3$, as well as the segments of the BIST register and the segment of the monitor.

Control signals in RSNs can be external or internal. If a control signal comes from outside of an RSN, it is referred to as an external control signal. If a signal comes from an update register of a configuration segment, it is called an internal control signal. The following control primitives are commonly used to build RSNs:

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

Each SIB (as shown in Fig. 2a) can be represented as a combination of a scan segment and a scan multiplexer, as shown in Fig. 2b. The underlying segment is only selected if the SIB is asserted. If the SIB is de-asserted, the segment is bypassed.

A small part of an RSN is shown in Fig. 3 with more details. A post-SIB (shown as SIB in Fig. 3) includes or excludes the remainder of the RSN from an activated path depending on the control assignments. This RSN is used as a running example for the remainder of the article.

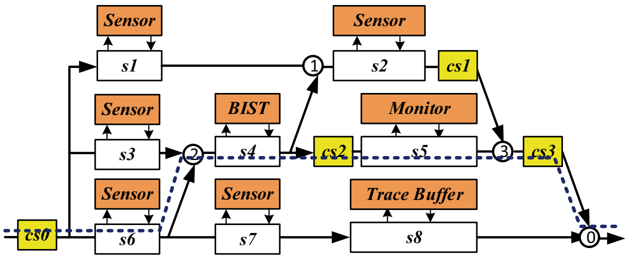

Fig. 1 A Reconfigurable Scan Network accesses instruments

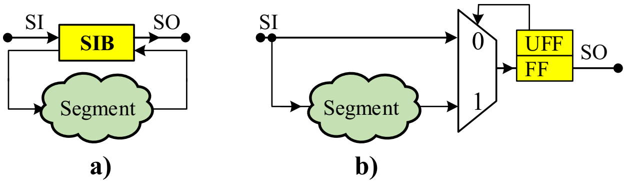

Fig. 2 SIB transformation for a post-SIB (configuration segment is located after the multiplexer) **a**) High-level representation **b**) Detailed structure

In an RSN, *scan segments* are the scan primitives, which shift the data through the RSN, as well as capture and update the data through a parallel interface. Each scan segment contains a shift register and an optional update register, as shown in Fig. 4. The following types of scan segments exist:

- *Data Segments* are scan segments, which access the instruments through a parallel interface. In data segments, the update registers serve as intermediate storage for the information, which is provided to the instruments.
- *Configuration Segments* are scan segments, where the information from the update registers is used to drive the internal control signals. The state of configuration segments defines the scan configuration.

Each access to an RSN can be represented as a transaction. This transaction is commonly referred to as a Capture-Shift-Update (CSU) operation [1]. The data is captured from the instruments into the shift registers during the *capture*-phase. During the *shift*-phase, the new data is shifted-in from a primary scan input through an active scan path, while the old data is shifted-out towards the scan output. Finally,

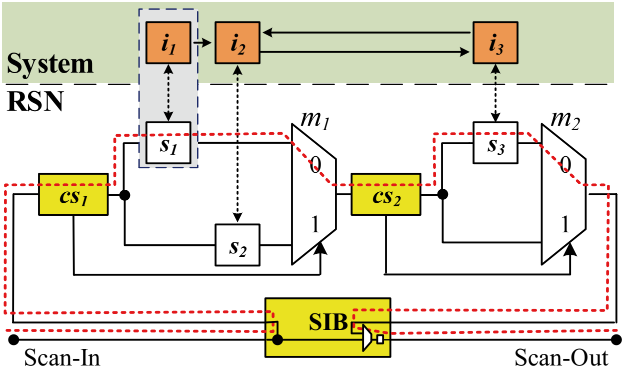

Fig. 3 Implementation of a small RSN part. $i_1 - i_3$ are the instruments; cs_1, cs_2 are the configuration segments; $s_1 - s_3$ are the scan segments; m_1, m_2 are the scan multiplexers. The SIB is asserted and the multiplexers m_1 and m_2 are driven with the value "0". The ASP is shown in red

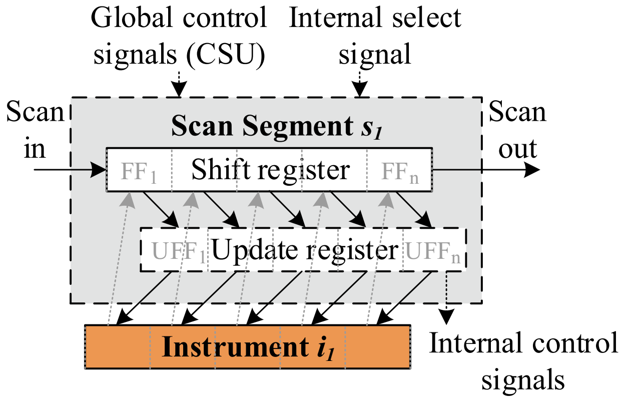

Fig. 4 Scan segment internals

during the *update*-phase, the newly shifted-in data is clocked into the update registers of the scan segments. The external global control signals control the CSU operations and bring an RSN into a known deterministic *reset* state if required. The data at the scan input port comes from an access interface, which can be either a Test Access Port (TAP) Controller, or an alternate functional or non-functional interface, as specified in the IEEE P1687.1 standard proposal [30]. The collected data might be processed and evaluated offline or online using on-chip computing, edge computing, or even in the cloud.

Each scan segment consists of one or multiple scan cells. A single scan cell $Cell_j$ consists of a shift flip-flop FF_j (a part of a shift register in Fig. 4), an optional update flip-flop UFF_j (a part of an update register), as well as of a few multiplexers to control CSU-operations. A gate-level structure of a single scan cell is shown in Fig. 5.

In a scan cell, the following paths are activated within a CSU operation:

- *A Shift Path* starts at the scan input (*SI*) and ends at the scan output (*SO*) of a cell. It contains two multiplexers (the shift multiplexer M_1 and the capture multiplexer M_2) and an internal data path of the shift flip-flop. During the shift-phase of a CSU operation, the shift multiplexer M_1 propagates the data from the scan input through the capture multiplexer M_2 and the shift flip-flop towards the scan output.
- *An Update Path* starts at the scan flip-flop's output and leads to the data output Q. This output may be connected to an instrument or drive RSN-internal control signals. The path comprises the update multiplexer M_3 and the internal data path of the update flip-flop. During the update phase, the update multiplexer M_3 propagates the data from the update flip-flop to the output Q.
- *A Capture Path* starts at the data input D, traverses the capture multiplexer M_2 and the data path of the scan flip-flop. During the capture phase, the capture multiplexer M_2 propagates the data from the instrument into the shift flip-flop.

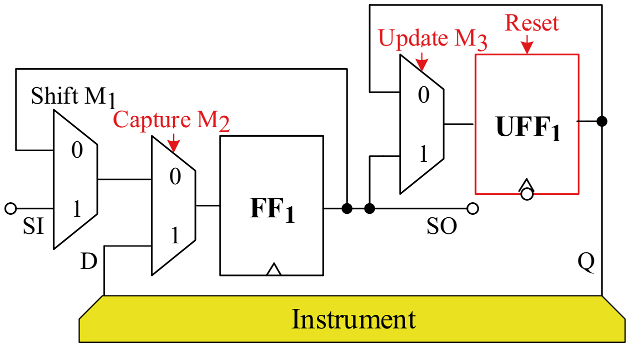

Fig. 5 Scan cell accesses an instrument through the data input D and the data output Q

2.2 RSN Model

An RSN is modeled as a directed graph $G := (V, E)$, where V is the vertex set, and E is the edge set. Each vertex corresponds to a scan primitive, a primary scan input or output, or represents a fanout stem f_i, as shown in Fig. 6 for the example from Fig. 3. Each edge models a direct connectivity between the vertices.

A source of an RSN graph is a vertex, which has only outgoing edges, while a sink has only incoming edges. We assume a single source $SI \in V$, and a single sink $SO \in V$. If the modeled RSN has multiple scan inputs, then the corresponding vertices are connected to a single pseudo-primary source vertex. The same logic is valid for RSNs with multiple scan outputs. Control scan primitives, i.e., multi-input scan multiplexers, SIBs, are modeled as a combination of one or multiple scan segments and one or multiple two-input scan multiplexers. The following relations are determined for an RSN graph:

- *Structural reachability:* A vertex m_j is structurally reachable from a vertex m_i, if at least one path exists from m_i to m_j.
- *Reconvergence vertex* [24]: A vertex m_j is a reconvergence vertex of the vertex m_i, if there are two paths p_1, p_2 with the corresponding vertex sets $V(p_1)$ and $V(p_2)$ such

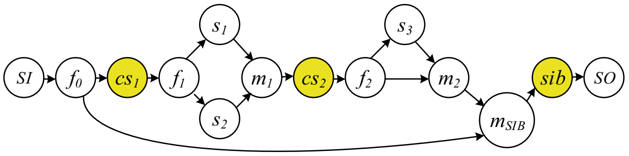

Fig. 6 RSN graph model

that $V(p_1) \cap V(p_2) = \{m_i, m_j\}$, m_i is the source of both p_1 and p_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.
- *A reconvergence region* of a vertex m_i includes all the vertices, which are reachable from this vertex and also reach its closing reconvergence.

Example: *In Fig.* 6, the reconvergency region of the vertex f_1 includes the vertices s_1 and s_2, and the vertex m_1 is a closing reconvergence.

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

Communication to the attached instruments and generation of internal control signals can be affected by faults in the update flip-flops, as well as the multiplexer M_2 and M_3. Possible fault locations are shown in Fig. 5 with a red color and explained below:

- *A capture multiplexer M_2 and data path of an update flip-flop:* A timing violation affecting the update flip-flop or a fault at the capture multiplexer M_2 may corrupt writing the data to the instrument during the update phase.
- *An update multiplexer M_3*: If the update multiplexer M_3 is faulty, it may prevent from reading correct data from the instrument during the capture phase.
- *A reset line of an update flip-flop:* If the reset line of an update register is affected by a stuck-at-0 fault, it may not be possible to reset its value into an initial known state.

In general, controllability and observability shall never be exercised through the instrument. Any attempt to do so would require a bespoke and hence a non-scalable DfT solution. At the same time, the logic around an update register can only be observed via the instrument. That makes the value of the signal Q unobservable for a test. Similarly, the value of D is uncontrollable, since it fully depends on the value of the instrument. This makes the faults at the capture multiplexer M_2 and the update multiplexer M_3, as well as the faults affecting the update flip-flops, in general not testable.

Example: *In Fig.* 3, *a fault may 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 provided to 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 [6]:

- *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 certain 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 the scan multiplexer m_1 from Fig.* 3 *is affected by a stuck-at-1 fault. Due to this fault, the scan segment s_1 becomes inaccessible. The latter 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 shift flip-flops. These violations may prevent correct data from being latched into the flip-flops while shifting.

Example: *If the shift flip-flop of the scan segment s_2 from Fig.* 3 *has a setup-time violation, the data is not properly latched into this flip-flop. The propagation through the activated path which traverses the scan segment s_2 is affected.*

3 State of the Art

Testing RSNs has been extensively studied in recent years. This section summarizes the existing methods to test RSNs with respect to the fault locations discussed above.

3.1 Test of Scan Interfaces

In [18], the primitives are tested, which are located at the capture- and the update-paths of scan segments. Read and write operations with opposite values are performed for each segment on the active scan path. However, as discussed above, it is not possible to test the primitives located at the interfaces independently from the values stored at the connected instruments. In realistic designs, the value of the instrument, may not be controllable or observable. Moreover, the existing test methods do not consider testing reset lines of update registers.

3.2 Test of Control Primitives

Numerous works in the past have presented methods to detect faults in control primitives and control lines. In [18], the conditions for activating faults, which may alter or break an activated scan path, are formally analyzed with a deterministic test pattern generator. The generator tests 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. [4] presents the first method to test those update registers, which guide the operation of control primitives. In [6], the control primitives themselves are targeted. A method is presented for smaller RSN designs to minimize the test application time while detecting faults in the control primitives. In [5], the scalability of the test method above has been significantly improved by presenting a scalable evolutionary heuristic. In [11], the test method above has been used as a basis for an efficient diagnostic procedure for permanent faults in the control logic. An approach from [11] performs access time optimization for RSNs located in multiple power domains, and [12] enhances the method above in terms of scalability. In [10], a post-silicon validation technique is presented, which identifies possible mismatches between the specification and the actual silicon implementation of RSNs. In [9], this method has been improved to consider equivalence between the structural description of an RSN and its Register Transfer Level implementation using simulation.

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 [1, 4–6, 9, 10, 18, 32]. However, if a 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 [10], the first systematic solution to detect them during the test has been presented in [23] which is extended in the article at hand.

3.3 Test of Scan Segments

Cantoro et al. [4] presents a method to test the scan shift logic of a particular scan segment. First, an active scan path is configured to select the target segment. Next, faults are tested 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 segments on the scan path are fault-free. Otherwise, there is a fault. For "stuck-at-faults", flush sequences, such as 01100 or 10011, are used. 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.

4 Overview of the Developed DfT Scheme

As discussed above, specifics of some RSN structures may affect the fault coverage. In Fig. 7, some examples of the testability issues are presented, which would arise for the RSN example from Fig. 3 if the existing test methods are applied.

1. *Undetected fault affecting an update register of* s_3: If the update 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 the value in the instrument i_3 is directly observable, which is not always true.
2. *Faulty Reset Line*: The existing methods do not guarantee to detect faults affecting the reset values of the update registers.
3. *Undetected 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. 7 instead of the intended path in Fig. 3. Since both paths have the same length, the fault at m_1 would remain undetected, and the data from the instrument i_2 would appear at scan output instead of the data from the instrument i_1.
4. *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 with the functional operation.

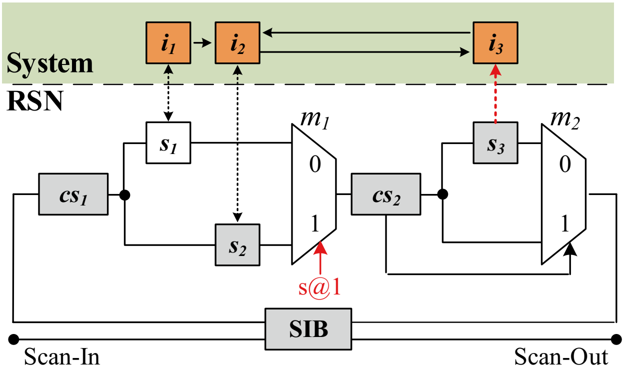

Fig. 7 Testability issues in the RSN example from Fig. 3

The remainder of this article presents a complete design-for-test (DfT) solution for RSNs which overcomes the above mentioned limitations of existing schemes.

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 apply to large arbitrary RSN designs.
- **Compactness:** Test sequence must be able 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 Testability of the Scan Interfaces

This section discusses the testability enhancement of scan interfaces between the scan segments and the instruments. First, the problem is formulated for the fault locations, which cannot be tested with the existing methods. Next, a DfT enhancement is presented to significantly increase the coverage of the faults at the scan interfaces of all data scan segments without corrupting the data stored in the instruments. As a result, the scan cell internal multiplexers (M_2 and M_3 in Fig. 5) and the update registers must become testable.

5.1 DfT Enhancement

The testability of the scan interfaces is improved by significantly increasing the observability of the update registers. The test of the multiplexers M_2 and M_3 is decoupled 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 output port by using conventional test methods.

The test of scan interfaces to the instruments is enabled by augmenting the initial scan cell structure (Fig. 5) with an additional feedback loop between the update flip-flop and the shift flip-flop, as shown with green color in Fig. 8.

The DfT structure provides direct visibility of the update flip-flop without requiring knowledge about or control over the connected instrument. The feedback loop propagates the value stored in the update flip-flop into a shift flip-flop. This data is then shifted through an activated scan path, such that the value of the update 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 [15]. The feedback loop can be described using 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.

The remainder of this section discusses how the DfT scheme is applied to test 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.

5.1.1 Testability Enhancement for the Scan Interfaces

In an enhanced scan cell (Fig. 8), 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. 8) and the initial paths through a scan cell by applying the following steps:

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

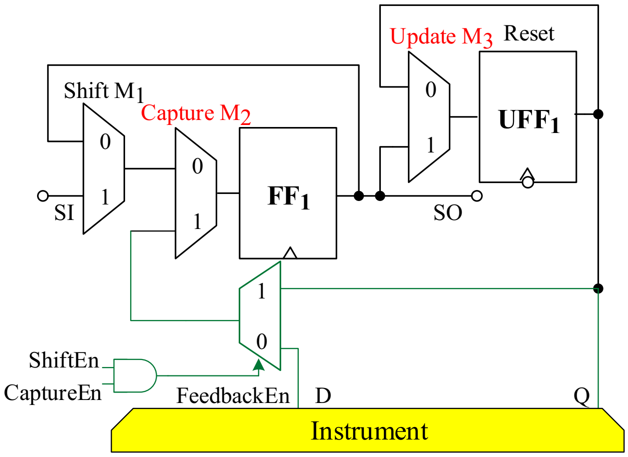

Fig. 8 Scan cell with a DfT Enhancement (in green). The additional scan multiplexer allows to propagate the data from the update flip-flop to the shift flip-flop

2. 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.
3. Finally, the data at the scan output of the scan segment is further propagated through an RSN by applying regular CSU operations.

5.1.2 Testability Enhancement for the Reset Line

The reset lines of update 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 update flip-flops into the shift flip-flops through the 1-branch of the feedback multiplexer, as shown in Fig. 8. This value is propagated towards the global scan out 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 update registers through the feedback loop and shifted out of the RSN. The shifted-out sequences for reset and non-reset states are compared to test a fault. Finally, a global reset signal is applied again to bring the RSN into its initial state.

5.1.3 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 $FeedbackEn$.

6 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. If a 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 affecting the control primitives are testable. As a result, 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 provide a scalable method for so-called series-parallel RSNs defined below. Finally, we show how can an arbitrary RSN be modeled as a series-parallel one.

6.1 Testability Concept

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

Definition 1 An active scan path $path_l$ is called to be *"single fault reachable"* from another path $path_k$, if and only if there is a single fault f which activates the path $path_l$ instead of $path_k$ 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. 9, 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.*

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

Definition 2 A single fault f affecting the RSN control primitives 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. Otherwise, a fault is categorized as *"undetectable by a path length (UDT-PL)"*, since the existence of such a test sequence is not guaranteed.

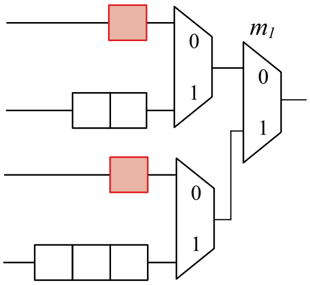

Fig. 9 Testability concept example

Example: *In Fig. 9, 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 at least one path length exists, 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 affecting the RSN control primitives, which may not be *detectable by differences in a path length*, and to resolve such untestable spots via resynthesis.

6.2 Series-Parallel RSN Model

In the following, the testability analysis is extended for large RSN designs by applying a divide-and-conquer algorithm on so-called series-parallel RSN models.

Definition 3 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 three statements is true:

1. G is an elementary series-parallel graph with $V = \{sc, si\}$; $E = \{(sc, si)\}$
2. G is a parallel composition of two series-parallel graphs $G_1 := (V_1, E_1), G_2 := (V_2, E_2)$:

$$\begin{aligned} V &:= V_1 \cup V_2 \\ E &:= E_1 \cup E_2 \end{aligned} \tag{1}$$

$$\begin{aligned} sc &:= sc_1 = sc_2 \\ si &:= si_1 = si_2 \\ V_1 \cap V_2 &= \{sc, si\} \end{aligned} \tag{2}$$

   sc_j and si_j are sources and sinks of G_j; $j = 1, 2$.
3. G is a series composition of two series-parallel graphs:

$$\begin{aligned} V &:= V_1 \cup V_2 \\ E &:= E_1 \cup E_2 \end{aligned} \tag{3}$$

$$\begin{aligned} sc &:= sc_1 \\ si &:= si_2 \\ si_1 &= sc_2 \end{aligned} \tag{4}$$

Any directed graph, which does not fulfill the conditions above is referred to as a *non-series-parallel graph*. Fig. 10a shows an example of a series-parallel graph, and Fig. 10b shows an example without the series-parallel property.

The hierarchical relations are stored in a binary decomposition tree, as shown in Fig. 11 for the running example, where the vertices corresponding to the multiplexers are located higher in the hierarchy than the vertices in their reconvergence regions. The leaves of the decomposition tree represent the scan primitives, while the intermediate vertices define, whether the subgraphs are connected in parallel, as shown with the green "P" vertices in Fig. 11, or in series, as shown with the blue "S" vertices.

Example: *Figure 11 shows a binary decomposition tree for the graph from Fig. 6. The tree is constructed bottom-up. First, the vertices s_1 and s_2 are connected in parallel via the vertex m_1, as shown with the* "P/m_1" *vertex in Fig. 11. Next, this sub-RSN is connected in series with the vertex cs_1 and then with the vertex cs_2. The same logic is applied to generate the remainder of the tree. The tree generation continues until the vertex modeling the configuration bit of the SIB is connected in series, as shown with the top-level "S" vertex of Fig. 11.*

In the following, we first present the initial testability analysis and resynthesis for the case of series-parallel RSNs. The developed approach processes large RSN designs in a scalable way. Finally, we show how can the developed methods be applied to arbitrary RSNs.

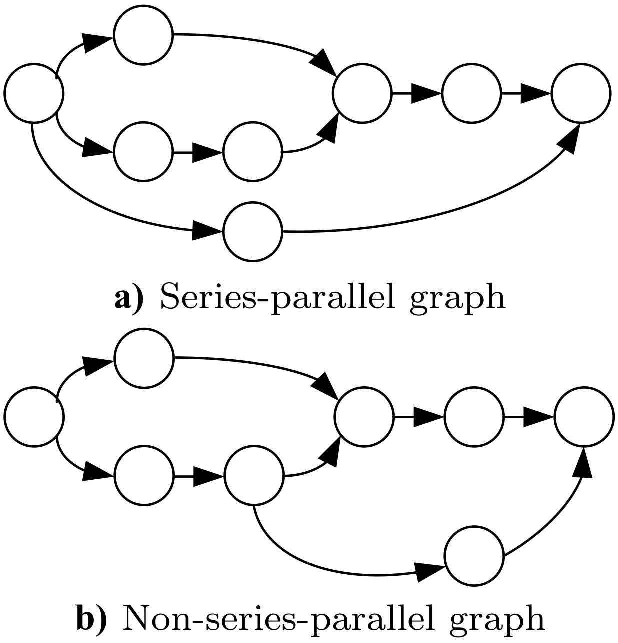

Fig. 10 Series-parallel property examples **a**) Series-parallel graph **b**) Non-series-parallel graph

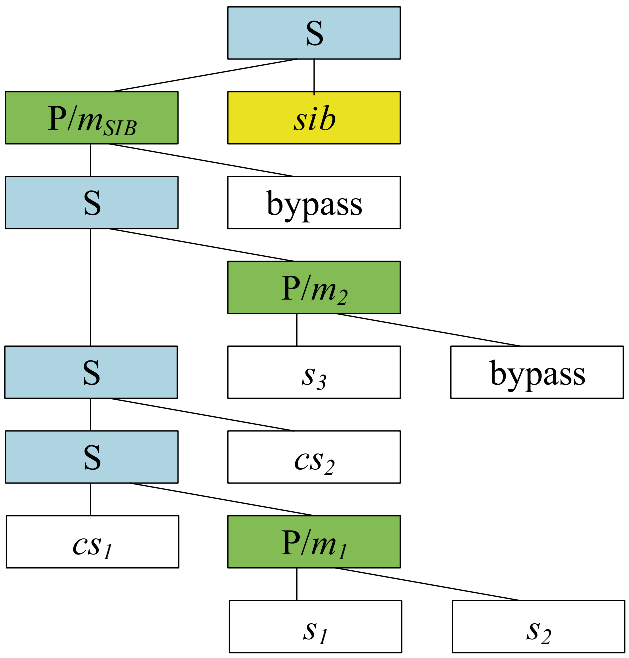

Fig. 11 Binary decomposition tree for the graph in Fig. 6

6.3 Testability Analysis of Series-Parallel RSNs

In this section, a divide-and-conquer approach is formulated to process the series-parallel graph of an RSN in a bottom-up manner. The analysis starts with elementary graph structures, such as parallel and series connections between the vertices of the RSN graph.

For the vertices connected in parallel, the testability concept from Section 6.1 is applied. The testability of the vertices connected in series does not depend on 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. The computation then proceeds with the analysis of bigger subgraphs until the whole RSN graph is processed. The remainder of this subsection presents the details of the testability analysis implementation.

Let the set $paths_j$ be the set of paths through a subgraph G_j, where each path $path_l$ has a length $pathLen_l$ and is activated if the path activation conditions $conds_l$ 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.

$$L := \{pathLen_1 + pathLen_2 \mid pathLen_1 \in L_1, pathLen_2 \in L_2\} \tag{5}$$

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

$$cond(path) := cond(path_1) \wedge cond(path_2) \tag{6}$$

A fault f in G_1, which is detectable by an altered path length, leads to a changed length of at least one path $path_1$ through G_1:

$$pathLen_1^f := pathLen_1 + \lambda \tag{7}$$

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:

$$pathLen^f := [pathLen_1 + \lambda] + pathLen_2 \tag{8}$$

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:

$$L := L_1 \cup L_2 \tag{9}$$

The sets of path lengths should not be intersecting:

$$L_1 \cap L_2 = \emptyset \tag{10}$$

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 $path_1$ and $path_2$ through G, such that $path_2$ is single fault reachable from $path_1$ or vice versa, the information about the differences between the corresponding path lengths is saved.

The paths through an RSN are analyzed recursively with a binary decomposition tree. Each time, the computation considers a series or a parallel composition of subgraphs. The initial computation starts with the leftmost leaf of the tree. 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. 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, until the whole RSN is analyzed.

If all the target faults are detectable by an altered path length, then the RSN is already testable. Then the testability-enhancing resynthesis is not needed for this RSN. Otherwise, if the RSN is not testable, the information about the control primitives with undetectable faults is saved and used for resynthesis. The saved information also includes the possible differences of the partial path lengths.

Example: *Given the decomposition tree from Fig. 11, the computation starts at the configuration 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.*

6.4 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 not "single fault reachable". If $m \notin Diff$, we can insert m flipflops in front of the 1-inputs of the multiplexers which leads to distinct path lengths. If $-m \notin Diff$, we can put m scan cells in front of the 0-input, but not at the 1-input.

Example: *Consider the example from Fig. 9 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. 12.*

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. Each vertex 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 affecting the RSN control primitives are detectable by a changed path length.

6.5 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 in Section 6.5.1 below allows us 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 6.5.2. After an equivalent series-parallel representation of a non-series-parallel RSN graph is constructed, this generated representation is used for performing the testability analysis and the automated resynthesis presented above.

6.5.1 Validation of the Series-Parallel Property

To check whether a specific RSN graph is series-parallel, a few simple checks are applied first. Then a reduction algorithm based on [33] 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 to 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 non-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 [20].

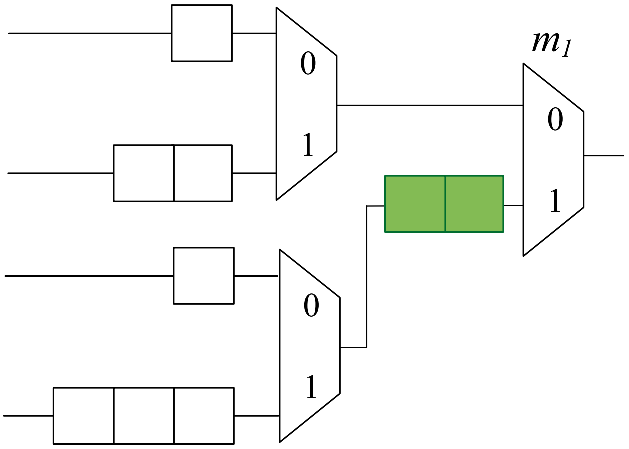

Fig. 12 Resynthesis example: The testability issue from Fig. 9 is resolved by inserting two scan cells

- *Main Flow* The main flow of the check follows the well-known reduction algorithm from [33]. 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 [7] allows to apply reductions in an arbitrary order to validate the series-parallel property.

6.5.2 Transformation into a Series-Parallel Graph

To build a series-parallel equivalent representation of a non-series-parallel graph region, a minimized number of additional virtual vertices is added into the initial RSN graph [17]. Since the virtual changes are reverted in the resynthesis phase, additional hardware overhead is not needed to transform the RSN graph into a series-parallel representation.

In the RSN graph, the fanout stems are identified, which prevent the RSN graph from being series-parallel. Any fanout stem f_{viol} in the stem region of another fanout stem f_{init}, 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 fanout 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 processing order, are selected in a topological order of the RSN graph, which starts at the scan input port. The fanout stems located closer to a primary scan-in vertex are processed first, followed by the fanout stems in their stem region. Each time, the computation either goes deeper in the hierarchy or moves forward to a succeeding fanout stem.

Example: *In Fig. 13, 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 two vertex representation will not be achieved.*

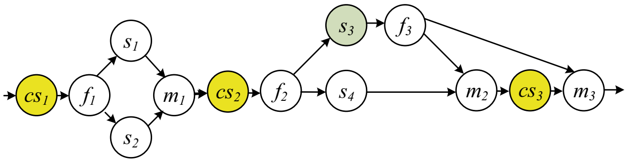

Fig. 13 Non series-parallel graph

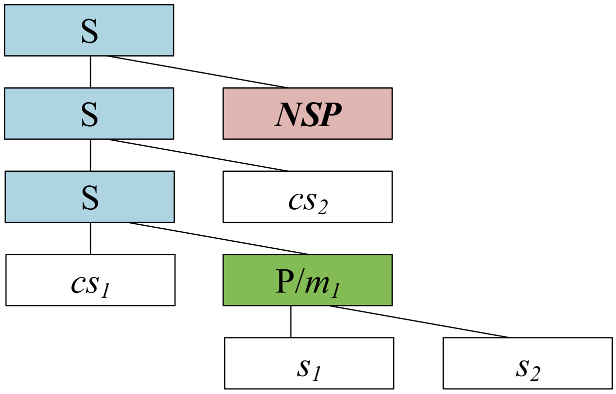

Fig. 14 Binary decomposition tree for the non-series-parallel RSN graph in Fig. 13

The decomposition tree for the resulting structure is shown in Fig. 14. In Fig. 15, an NSP region is transformed into a series-parallel form by duplicating the vertex s_3 and the fan-out stem f_3.These changes are virtual, are only used to achieve a scalable computation flow and will be reverted after the resynthesis, which works as follows.

Assume that in Fig. 13 the scan segments corresponding to the vertices s_4 and s_3 have the same length. Then the duplicated vertex s_{3c} in the series-parallel representation in Fig. 15 would also have the same length as s_4. As a result, the fault affecting the vertex m_2 will be undetectable by an altered path length. To resolve the testability problem, an additional scan cell will be added to the RSN. This change is not virtual and is not reverted, since it enhances the testability of the network.

The resulting binary decomposition tree, as shown in Fig. 16, 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 testability, as discussed above.

7 Test of Scan Segments

The method presented above ensures that the faults affecting the control primitives are testable. In this section, the test of scan segments is considered. In contrast to the existing

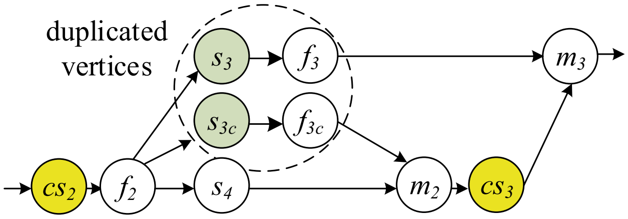

Fig. 15 Transformed subgraph of the non-series-parallel graph from Fig. 13

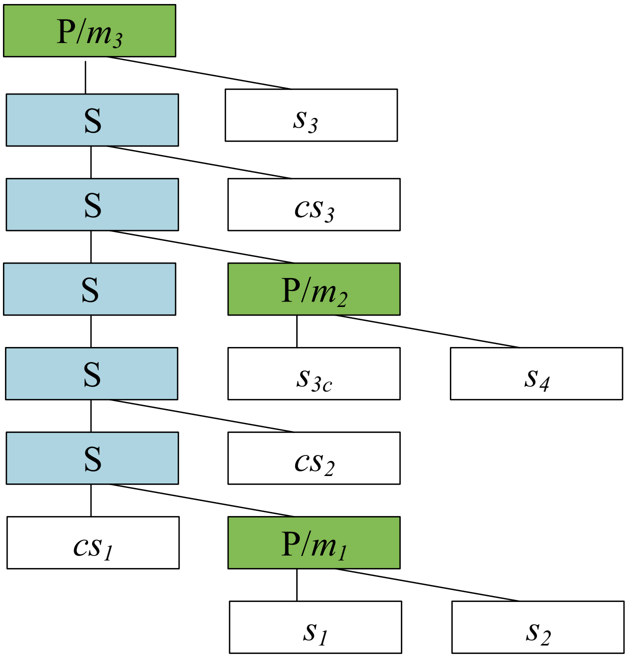

Fig. 16 Binary decomposition tree for a series-parallel representation of the non-series-parallel graph from Fig. 13

schemes in Section 3, the test of scan segments is applied concurrently with instrument access. A compact built-in self-test structure is added to the RSN and is used to generate a short test presequence. This presequence is augmented with a workload sequence, shifted into 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 7.1. An example implementation of a concurrent BIST structure for RSNs is shown in Section 7.2.

7.1 Test Pattern Generation

Each complete test sequence (Fig. 17a) includes a workload sequence W and a flush test sequence T. Workload sequences access RSNs and are generated as further discussed in Section 8.2. 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 $\overline{T} =< \overline{t}_{n-1}, ...\overline{t}_0 >$ is one as well.

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 automated test equipment (ATE) together with the workload sequences or generated on-site, as discussed below in Section 7.2.

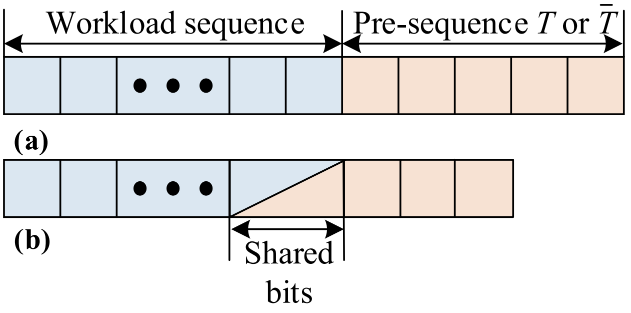

Fig. 17 Test sequence construction **a**) workload sequence is augmented with a flush sequence **b**) bit sharing mechanism

Fig. 17b represents the bit sharing mechanism, which is used for merging the workload sequence with the flush presequence. 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.

7.2 Test Pattern Application

ROSTI (RSN Online/Offline Self-Test Infrastructure) is a self-test structure for RSNs to generate test sequences and attach them to the workload sequences. Its structure is shown in Fig. 18 and includes a test sequence generator (TSG), an acceptor, and a controller. ROSTI is placed between the RSN and the TAP controller.

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 presequence 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.

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. The same applies

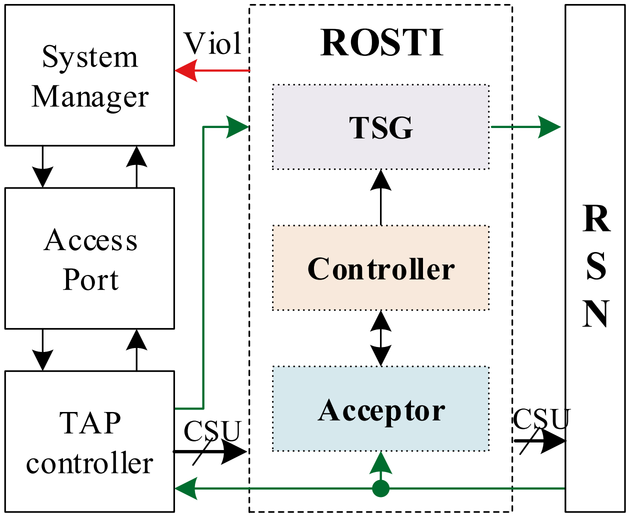

Fig. 18 RSN Online/Offline Self-Test Infrastructure (ROSTI) structure

to 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 major 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 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. It allows deciding which flush test sequence ("01100" or "10011") to use in a given test sequence. The acceptor is constructed as a finite state machine, which consists of a few flip-flops and a few logic gates. The presented acceptor is independent of the length of the ASP. 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 the 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 toward the acceptor to allow correct test response comparison.

The P1687.1 standard proposal [30], which is also discussed in [8], suggests using access mechanisms rather than just a JTAG Test Access Port (TAP) controller to access an RSN. [21] presents a hardware module to perform online retargeting block which is used as a part of an access mechanism. In [27], a scan encryption module is implemented as a part of a custom access mechanism. TIn the paper at hand, the developed self-test block represents an access interface together with a TAP controller and an access port, which enables RSN self-test and is also in line with the P1687.1 standard proposal.

8 Test Integration

This section discusses the integration of the presented DfT scheme into the RSN-under-test. First, a summary of the necessary changes to the RSN structure is presented followed by some details about test sequence construction for the enhanced RSN. Finally, the application of the developed DfT scheme throughout the lifetime is discussed.

8.1 Changes to the RSN Structure

The developed DfT scheme for scan interfaces, control primitives, and segments is integrated during the design phase. A summary about the testability enhancements is provided below.

1. **Scan Interface Observability Enhancement** (see Section 5): The design-for-test scheme is integrated to increase the observability of the update registers. As a result, the faults in the capture- and update-circuity of the scan segments become detectable, and the interfaces to the instruments (including the interface of s_3 in Fig. 7) can be tested. The reset functionality of the update flip-flops is now also testable.
2. **Control Primitives Testability Enhancement** (see Section 6): The RSN structure is analyzed to check whether

any single fault affecting the RSN control primitives is undetectable by a changed path length. It implies, that if the path length is correct, then the correct path is activated through the RSN. 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.

3. **Scan Segment Test Enhancement** (see Section 7): The described compact BIST hardware is integrated into the RSN. It allows testing the shift logic of the scan segments concurrently. A fault affecting a scan segment is detected with the help of a flush test presequence, as soon as a path through this segment is activated.

In Fig. 19, the example from Fig. 7 is enhanced with the required DfT changes above. In the resulting RSN, all the testability issues are resolved.

The developed scheme is intended to support the offline test of RSNs. It not only tests the control primitives and the interfaces to the instruments but also examines the shift logic of those scan segments which are included in the currently configured Active Scan Path. Although an Automated Test Equipment (ATE) can also handle testing the scan segments, it could be costly since it requires an extra step. ROSTI automatically generates and compares flush sequences, and, in the presence of ROSTI, an ATE will only need to examine one violation signal. After the presented DfT scheme is integrated, ROSTI can be reused to test the shift logic of the scan segments on the activated scan path online concurrently to the functional workload. If a fault is detected by ROSTI, the information about it can be further reused to support the operation of fault-tolerant or error-resilient networks [2].

8.2 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. Test compression and scheduling algorithms are beyond of the scope of this paper. A set of test sequences can be generated automatically to cover the whole RSN structure with a minimized test application time, as in [1, 5, 12].

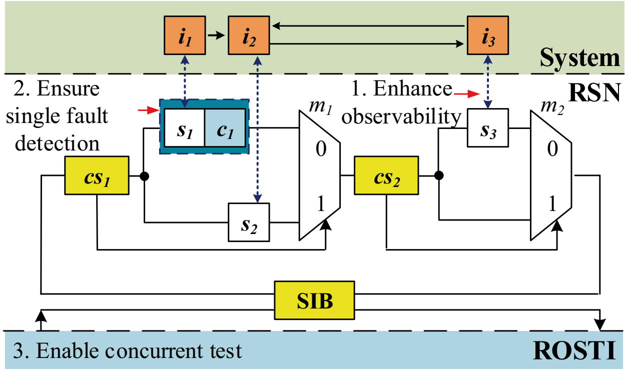

Fig. 19 RSN example from Fig. 7 is enhanced by using the developed DfT Scheme

To keep this paper self-contained, a test sequence generation process is briefly summarized below. Unlike the conventional structural test, the test of RSNs requires multiple reconfigurations. Therefore, workload sequences generated by a TSG are of two types:

- *Access Sequences* configure a desired path through the RSN by switching scan multiplexers, and opening and closing SIBs. Those sequences are generated by using a Test Sequence Generation algorithm. In this work, the TSG is based on the retargeting engine, which has been first published in [1]. It ensures that the scan primitives of an RSN are covered by activating a minimized number of activated scan paths.
- *Workload Test Sequences* test the scan primitives which are included in the activated path. They are usually based on flush test sequences and are applied after a desired active scan path is configured.

Each workload test sequence may include 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, the flush test sequence will be shifted-out unchanged in the fault-free case. 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 single faults affecting the RSN control primitives. The same applies to the single flip-flop transparency faults in the shift registers, since they reduce the length of the activated path by one shift cycle. Similarly, an access sequence contains a sequence to retarget an RSN and a flush test sequence T to test the shift logic on the path.

9 Experimental Results

The complete design-for-test method is implemented in the framework *eda1687*, which has first been presented in [1]. It uses Instrument Connectivity Language (ICL) descriptions of RSNs as input for test generation and generates Hardware Description Language (HDL) descriptions for gate-level synthesis.

The experiments have been conducted on a CPU Intel(R) Xeon(R) W-2125 CPU at 4.00GHz with 132 GB of main memory. The remainder of the section summarizes the experiments

Table 1 Control Primitives Testability Enhancement

(1) **Design**	(2) # Total Faults	(3) # Undetected Faults	(4) # Total Cells	(5) # Added Cells	(6) Runtime [s]
BasicSCB	40	8	176	4	1.0
Mingle	52	16	270	8	1.2
TreeBalanced	200	12	5,581	6	2.1

for the individual DfT enhancements. Next, the results are provided for the complete DfT method, which considers test integration. In the experiments, stuck-at faults affecting scan interfaces, scan segments, and scan multiplexers are considered.

9.1 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.

9.2 Control Primitives

For any RSN design, the developed DfT method ensures that the RSN is testable for single faults in the control primitives. If all faults affecting the control primitives 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 for single faults affecting the RSN control primitives and is thereby one of the major contributions of this article. To ensure fault detection, the lengths of a minor number of scan chains may be slightly increased.

The scalability and the effectiveness of the developed method have been proven using the benchmarks from the ITC'2016 [31] and the DATE'2019 benchmark sets [28]. As shown in Table 1, for the benchmarks *TreeBalanced, Mingle, BasicSCB* from the ITC'2016 [31]set, the testability analysis identified single stuck-at faults affecting the RSN control primitives, which are undetectable by an altered path length (Column 3). The total number of faults is reported in Column 2. To ensure fault detection, a minor number of scan cells (Column 5), has been added to the initial RSN structure. This number is negligible compared to the total number of faults in the benchmark (Column 4). Thanks to the scalable algorithm, the runtime is acceptable even for the most time-consuming benchmarks (Column 6).

9.3 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. To test ROSTI itself, a commercial tool has been used to perform test pattern generation with a 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 of the RSN. The gate-level fault coverage for stuck-at-faults is determined with a commercial sequential stuck-at-fault simulator.

9.4 Complete DfT Method

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 shift logic, the benchmarks have been constructed with the help of the ITC'02 SoC (System-on-a-Chip) benchmark set [25]. The characteristics of the benchmarks are presented in Table 2. In Column 2 the number of hierarchical levels, and the number of scan multiplexers are given, followed by the number of scan segments in Column 4 and the number of scan cells in Column 5.

The experimental results for the developed scheme are shown in Table 3:

- *Integration of the Design-for-Test Scheme:*

Table 2 RSN Benchmark circuits

Design	#Hier. lvl	#Scan muxes	#Scan segs	#Scan cells	#ASPs
u226	2	59	99	1 457	615
d281	2	67	117	3 880	774
d695	2	178	335	8 407	2,385
h953	2	63	109	5 649	702
g1023	2	94	159	5 400	1,005
f2126	2	45	81	15 834	540
q12710	2	30	51	26 188	327
p34392	3	142	245	23 261	1,815

Table 3 Experimental results

	Control primitives	Runtime	Test Cost [#cycles]				Overhead[%]	Coverage [%]
(1)Design	(2) #faults	(3) [s]	(4) Inter.	(5) Segs.	(6) Sum	(7) Our	(8) with respect to the RSN	(9) with feedback
u226	118	0.2	15,536	22,647	38,183	17,996	0.69	93.65
d281	134	0.2	32,863	34,113	66,976	35,959	0.27	95.51
d695	356	0.3	84,026	116,234	200,260	93,566	0.19	92.17
h953	126	0.2	44,296	38,247	82,543	47,104	0.18	96.32
g1023	188	0.3	46,443	50,418	96,861	50,463	0.20	95.66
f2126	90	0.1	114,563	75,269	189,832	116,723	0.06	95.11
q12710	60	1.0	184,971	109,904	294,875	192,231	0.03	95.43
p34392	288	2.2	181,591	156,403	337,994	188.851	0.06	94.70

  - *Scan Interfaces:* Scan registers are enhanced by injecting a feedback line.
  - *Control Primitives:* The testability for single stuck-at faults in the control primitives is proven for all the benchmarks. As detailed in Section 8.4, for general RSNs the testability property is not guaranteed. The total number of faults affecting the control primitives 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:* The test cost in terms of the number of clock cycles for different test sequence sets is given in Columns 4-6. The details about the generated test sequence sets are provided below:

  - *Scan Interfaces and Control Primitives:* Access sequences configure a desired active scan path. They are generated as in [1] and cover all the scan segments and all the branches of scan multiplexers. Since any active scan path which includes a faulty control primitive is guaranteed to have a different length compared to a fault-free path, faults in control primitives are detectable. Workload test sequences are generated to test scan interfaces in the enhanced RSN as detailed in Section 5. The corresponding test cost in terms of clock cycles is provided in Column 4.
  - *Scan Segments:* To test those scan segments, which are located on the configured path, a test sequence is constructed of a workload test sequence to configure a desired path and a flush test sequence. ROSTI generates flush test sequences to test the shift logic of the selected scan segments. The test cost is given in Column 5.

Area overhead compared to the underlying RSN is given in Column 8 and is negligible. A commercial sequential stuck-at-fault simulator is used to determine the gate-level fault coverage. The fault coverage for RSN benchmarks with feedback lines in the scan segments is given in Column 9. Fault coverage is above 92.60% for nine benchmark circuits, and 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 the scan interface, a *complete fault coverage* is obtained for all the benchmarks.

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

10 Conclusion

In this paper, the first design-for-test scheme is presented, which allows for complete covering all the stuck-at faults in a Reconfigurable Scan Network. It 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 for significantly optimizing the size of the test sequence set.

The presented scheme is flexible for 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 methods for RSNs and is supposed to be used complementary to these schemes. The scheme is also flexible with respect to the access mechanisms, and can be controlled by the workload test patterns from an ATE, from the cloud, or even stored on-chip internally. The experimental results show that the presented scheme generates test sequences with complete fault coverage and reduced test cost. It is scalable with the increasing size and complexity of RSNs.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the German Research Foundation (DFG) under grant WU 245/17-2 (ACCESS) and partially supported by Advantest as part of the Graduate School "Intelligent Methods for Test and Reliability" (GS-IMTR) at the University of Stuttgart.

Data Availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflicts of Interest The authors declare that they have no conflict of interest or competing interests.

References

1. Baranowski R, Kochte MA, Wunderlich H-J (2015) Reconfigurable scan networks: modeling, verification, and optimal pattern generation. ACM Transactions on Design Automation of Electronic Systems (TODAES) 20(2):1–28
2. Brandhofer S, Kochte MA, Wunderlich H (2020) Synthesis of fault-tolerant reconfigurable scan networks, in Proc. Automation Test in Europe Exhibition (DATE), Mar, Design, pp 798–803
3. Cantoro R, Damljanovic A, Reorda MS, Squillero G (2018) A new technique to generate test sequences for reconfigurable scan networks in Proc. IEEE International Test Conference (ITC) pp 1–9
4. Cantoro R, Montazeri M, Reorda MS, Zadegan FG, Larsson E (2015) On the Testability of IEEE 1687 Networks, in Proc. IEEE Asian Test Symposium (ATS) pp 211–216
5. Cantoro R, San Paolo L, Sonza Reorda M, Squillero G (2018) An evolutionary technique for reducing the duration of reconfigurable scan network test, in Proc. IEEE International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS) pp 129–134
6. Cantoro R, Zadegan FG, Palena M, Pasini P, Larsson E, Reorda MS (2018) Test of reconfigurable modules in scan networks. IEEE Transactions on Computers (TC) 67(12):1806–1817
7. Church A, Rosser JB (1936) Some properties of conversion. Trans Am Math Soc 1(2):472-482
8. Crouch AL, Van Treuren BG, Rearick J (2020) P1687.1: Accessing Embedded 1687 Instruments using Alternate Device Interfaces other than JTAG, in Proc. IEEE European Test Symposium (ETS), 2020, pp 1–6
9. Damljanovic A, Jutman A, Portolan M, Sanchez E, Squillero G, Tsertov A (2019) Simulation-based equivalence checking between IEEE 1687 ICL and RTL, in Proc. IEEE International Test Conference (ITC) pp 1–8
10. Damljanovic A, Jutman A, Squillero G, Tsertov A (2019) Post-silicon validation of ieee 1687 reconfigurable scan networks, in Proc. IEEE European Test Symposium (ETS) pp 1–6
11. Habiby P, Huhn S, Drechsler R (2020) Power-aware Test Scheduling for IEEE 1687 Networks with Multiple Power Domains, in Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT) pp 1–6
12. Habiby P, Huhn S, Drechsler R (2021) Optimization-based test scheduling for IEEe 1687 multi-power domain networks using boolean satisfiability, in Proc. International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS) pp 1–4
13. Ibrahim AMY, Kerkhoff HG (2019) An On-chip IEEE 1687 network controller for reliability and functional safety management of system-on-chips, in Proc. IEEE International. Test Conference in Asia (ITC-Asia) pp 109–114
14. IEEE (2013) Standard for Test Access Port and Boundary-Scan Architecture IEEE Std. 1149.1-2013 (Revision of IEEE Std 1149.1-2001), pp 1–444
15. IEEE (2014) Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device, IEEE Std. 1687-2014, pp 1–283
16. IEEE Standard for System Test Access Management (STAM) to Enable Use of Sub-System Test Capabilities at Higher Architectural Levels, *IEEE Std. P2654*
17. Keller J, Gerhards R (2014) PEELSCHED: A simple and parallel scheduling algorithm for static taskgraphs, PARS: parallel-algorithmen, -rechnerstrukturen und -systemsoftware, vol. 28
18. Kochte MA, Baranowski R, Schaal M, Wunderlich H (2016) Test Strategies for Reconfigurable Scan Networks, in Proc. IEEE Asian Test Symposium (ATS) pp 113–118
19. Kochte MA, Wunderlich H-J (2018) Self-Test and Diagnosis for Self-Aware Systems. IEEE Des Test 35(5):7–18
20. Kunzmann A, Wunderlich H-J (1990) An analytical approach to the partial scan problem. J Electron Test Theory Appl (JETTA) 2(1):163–174
21. Larsson E, Murali P, Kumisbek G (2019) IEEE Std. P1687.1: Translator and Protocol, in Proc. IEEE International Test Conference (ITC) pp 1–10
22. Lee K-J, Breuer MA (1990) A universal test sequence for cmos scan registers, in Proc. Custom Integrated Circuits Conference pp 28.5/1–28.5/4
23. Lylina N, Wang C-H, Wunderlich H-J (2021) Testability-enhancing resynthesis of reconfigurable scan networks, in Proc. IEEE International Test Conference (ITC), Virtual App 1–10
24. Maamari F, Rajski J (1990) A method of fault simulation based on stem regions. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) 9(2):212–220
25. Marinissen EJ, Iyengar V, Chakrabarty K (2002) A set of benchmarks for modular testing of SOCs, in Proc. IEEE International Test Conference (ITC) pp 519–528
26. Portolan M, Rearick J, Keim M (2020) Linking chip, board, and system test via standards, in Proc. IEEE European Test Symposium (ETS) pp 1–8
27. Portolan M, Valea E, Maistri P, Natale GD (2022) Flexible and portable management of secure scan implementations exploiting p1687.1 extensions. IEEE Des Test 39(3):117–124
28. Raiola P, Thiemann B, Burchard J, Atteya A, Lylina N, Wunderlich H.-J, Becker B, Sauer M (2019) On secure data flow in reconfigurable scan networks, in Proc. Conference on Design, Automation Test in Europe (DATE), pp 1–6
29. Shibin K, Devadze S, Jutman A, Grabmann M, Pricken R (2017) Health Management for Self-Aware SoCs Based on IEEE 1687 Infrastructure. IEEE Des Test 34(6):27–35

30. Standard for the Application of Interfaces and Controllers to Access (1687) IJTAG Networks Embedded Within Semiconductor Devices, IEEE Std. P1687.1
31. Tsertov A, Jutman A, Devadze S, Reorda MS, Larsson E, Zadegan FG, Cantoro R, Montazeri M, Krenz-Baath R (2016) A suite of IEEE 1687 benchmark networks, in Proc. IEEE International Test Conference (ITC) pp 1–10
32. Ull D, Kochte M, Wunderlich H (2017) Structure-oriented test of reconfigurable scan networks, in Proc. IEEE Asian Test Symposium (ATS) pp 127–132
33. Valdes J, Tarjan RE, Lawler EL (1979) The recognition of series parallel digraphs, in Proc. Annual ACM Symp on Theory of Comput 1-12
34. Wang C-H, Lylina N, Atteya A, Hsieh T-Y, Wunderlich H-J (2021) Concurrent test of reconfigurable scan networks for self-aware systems, in Proc. IEEE International Symposium on On-Line Testing And Robust System Design (IOLTS), Virtual pp 1–7

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Natalia Lylina received the Master of Science (M.Sc.) double degree in computer science from Moscow Power Engineering Institute (National Research University), Russian Federation and Technical University of Ilmenau, Germany in 2017. Since 2017 she is with the Institute of Computer Architecture and Computer Engineering at the University of Stuttgart as a PhD student. She is a Student Member of IEEE. Her research interests include dependable systems, test and diagnosis infrastructure and reconfigurable scan networks.

Chih-Hao Wang received his B.Sc. and Ph.D. degree in electrical engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2013 and 2020, respectively. During 2019 to 2020, he was a visiting scholar of the Institute of Computer Architecture and Computer Engineering at the University of Stuttgart, Germany, and he is currently a postdoctoral researcher of the same institute. He is a Member of IEEE. His research interests include VLSI testability and reliability, concurrent error detection, and reconfigurable scan networks.

Hans-Joachim Wunderlich received the diploma degree in mathematics from the University of Freiburg, Germany, in 1981 and the Dr. rer. nat. (Ph.D. degree) from the University of Karlsruhe in 1986. Since 1991, he has been a full professor and from 2002 to 2018 served as the director of the Institute of Computer Architecture and Computer Engineering at the University of Stuttgart, Germany. He is a Life Fellow of IEEE. He has been associated editor of various international journals and program committee member of a variety of IEEE conferences on design and test of electronic systems. He has published 11 books and book chapters and around 300 reviewed scientific papers in journals and conferences. His research interests include test, reliability, fault tolerance and design automation of microelectronic systems.