SCAR: Security Compliance Analysis and Resynthesis of Reconfigurable Scan Networks

Lylina, Natalia; Wang, Chih-Hao; Wunderlich, Hans-Joachim

Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022

doi: https://dx.doi.org/10.1109/TCAD.2022.3158250

Abstract: Reconfigurable Scan Networks (RSNs) enable an efficient reliability management throughout the device lifetime. They can be used for controlling integrated instruments, such as aging monitors or built-in self-test (BIST) registers, as well as for collecting the evaluation results from them. At the same time, they may impose a security threat, since the additional connectivities introduced by the RSN can possibly be misused as a side-channel. This paper presents an approach for Security Compliance Analysis and Resynthesis (SCAR) of RSNs to integrate an RSN compliant with the security properties of the initial design. First, the reachability properties of the original design are accurately computed. The connectivities inside the RSN, which exceed the allowed connectivity of the initial design, are identified using the presented Security Compliance Analysis. Next, all violations are resolved by automated Resynthesis with a minimized number of structural changes. As a result of SCAR, any information leakage due to the RSN integration is prevented, while the accessibility of the instruments through the RSN is preserved. The approach is able to analyze complex control dependencies and obtains a compliant RSN even for the largest available benchmarks.

Preprint

SCAR: Security Compliance Analysis and Resynthesis of Reconfigurable Scan Networks

Natalia Lylina, *Student Member, IEEE,* Chih-Hao Wang, *Member, IEEE,*
Hans-Joachim Wunderlich, *Life Fellow, IEEE*

***Abstract*—Reconfigurable Scan Networks (RSNs) enable an efficient reliability management throughout the device lifetime. They can be used for controlling integrated instruments, such as aging monitors or built-in self-test (BIST) registers, as well as for collecting the evaluation results from them. At the same time, they may impose a security threat, since the additional connectivities introduced by the RSN can possibly be misused as a side-channel.**

This paper presents an approach for Security Compliance Analysis and Resynthesis (*SCAR*) of RSNs to integrate an RSN compliant with the security properties of the initial design. First, the reachability properties of the original design are accurately computed. The connectivities inside the RSN, which exceed the allowed connectivity of the initial design, are identified using the presented *Security Compliance Analysis*. Next, all violations are resolved by automated *Resynthesis* with a minimized number of structural changes. As a result of *SCAR*, any information leakage due to the RSN integration is prevented, while the accessibility of the instruments through the RSN is preserved. The approach is able to analyze complex control dependencies and obtains a compliant RSN even for the largest available benchmarks.

***Keywords*—Reconfigurable Scan Network; Secure DfT; Design validation; Synthesis; Integer Linear Programming; SAT**

I. Introduction

THE complexity of electronic circuits has been rapidly increasing throughout the last decades. In order to ensure fast yield bring-up and dependable operation of devices throughout the lifetime, many functional and non-functional instruments, such as sensors, aging monitors or BIST registers, are embedded. Reconfigurable Scan Networks, as standardized by IEEE Std. 1149.1 [1] and IEEE Std. 1687 [2], offer an efficient access to the instruments and enable a runtime reliability management [3], [4].

To ensure system-level security, a system designer thoroughly develops the connections inside a design 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. Attacks using DfT infrastructure, such as conventional JTAG scan chains are well-investigated in the literature [5]. A few real examples include but are not limited to the attacks on XBOX 360 [6], and on iPhones [7]. The additional connectivities in the design-under-test (DUT), introduced due to the RSN integration, might be exploited by an attacker as a side-channel to leak or manipulate sensitive data or alternate the system behavior [8]–[11]. The secure integration of RSNs is even more challenging compared to conventional scan chains [12], [13] due to the complex control dependencies [14]. Fully denying access to RSNs during the functional mode is not an option, since often the dependability instruments must be available online [15], [16].

It cannot be the responsibility of a DfT-integrator to repeat the entire security and threat analysis done by a system designer. However, it should be ensured that the original analysis and protection policy is not invalidated by the DfT-infrastructure. For this, the DfT-integrator has to follow both implicit and explicit security specifications of the design:

- *Implicit security 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.
- *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.

In this paper, we present a comprehensive solution for Security Compliance Analysis and Resynthesis (*SCAR*) of RSNs. The goal is to integrate an RSN in a way that these specifications are fulfilled and any additional information leakage through the RSN is eliminated. It allows to efficiently analyze the compliance of an RSN with the original security properties provided in specifications, and to automatically resynthesize the RSN with minimized effort, if any compliance violation is identified. The first exact approach to perform a security compliance analysis of RSNs has been published in [17]. In [18] the first automated resynthesis is presented to minimize the structural changes, which are required to securely integrate a given RSN into a DUT, even if the number of violations is high. The paper at hand extends these preliminary results by core contributions presented below:

- An efficient implementation of security compliance analysis in terms of runtime and memory consumption is presented.
- A complete graph-based approach is formulated to automatically integrate an RSN into a design, based on

N. Lylina, C.-H. Wang and H.-J. Wunderlich are with the Institute of Computer Architecture and Computer Engineering, University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany, Email: lylina@informatik.uni-stuttgart.de, wangco@informatik.uni-stuttgart.de and wu@informatik.uni-stuttgart.de. This paper combines and extends preliminary work published in ITC19 and ITC20.

Manuscript received MONTH DAY, YEAR; revised MONTH DAY, YEAR

the results of the security compliance analysis.
- Various optimization criteria, such as the minimized access latency and hardware overhead, are considered and compared for the automated resynthesis.

The remainder of this paper is organized as follows. Section II provides the background information and summarizes the existing solutions for securing the RSNs, and Section III provides details of the system modeling. Section IV shows an overview of the whole SCAR methodology. Then, in Section V, the security compliance analysis is presented. In Section VI, details for resolving the violations in RSNs are given, followed by the security validation procedure in Section VII. Finally, the experimental results in Section VIII demonstrate the efficiency and scalability of the approach.

II. Previous Work on Secure RSNs

A. Security Analysis and Specification

Security threats may exist due to unwanted access through the Test Access Port (TAP), but also due to connectivities between untrusted parts of an RSN and the pins of the chip [11].

Security properties of RSNs can be validated via simulation to check the data integrity by adding hash functions to shift sequences as in [9], and to provide an evidence about unauthorized access attempts. Machine learning-based techniques can be used to detect attacks on RSNs [10]. Security properties of RSNs, such as data confidentiality, can be verified by Craig interpolation [19]. A method from [20] 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 [21] or through a path [22], traversing both the DUT 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 DUT. This makes the existing analysis methods unscalable for large RSNs and realistic security specifications [23]. A high sequential depth of RSNs and complex control dependencies make the existing ways unfeasible to compute the functional reachability [24]–[26].

B. Security-Oriented Integration of Scan Chains

Multiple solutions exist to resolve the security violations, which have been identified by simulation or formal verification. In conventional scan chains, a separate ”secure” mode can be established to perform the confidentiality-critical computations [12]. The shifted data can be encrypted as in [27] or obfuscated as in [13]. An unauthorized transition from the functional mode to the test mode can be prevented as in [28] to mitigate the test mode attacks. For a comprehensive review on scan chain security refer to [29].

The flexibility of RSNs makes their secure integration more challenging compared to conventional scan chains. It is not an option to disable an RSN by fusing off the TAP port after performing the manufacturing test. This strategy, would make the further online usage of RSNs, such as in-field monitoring [15] or reliability enhancement [16], impossible. Encryption and obfuscation can secure the RSN access throughout the life-cycle [30], [31]. Fine-grained schemes, such as Locking Segment Insertion Bits (LSIBs) [32], Parallel LSIBs [33], and Secure SIBs (SSIBs) [34], extremely complicate an unauthorized access to specific RSN parts. [35] presents a fine-grained dynamic technique to protect RSNs. In [8], shadow registers and information-flow tracking logic is added to prevent data sniffing and alteration. In [11], [36], 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. Any additional connectivities, which exceed 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. In some cases, filters [37], [38] 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. The existing approaches [21], [22] resolve the violating connectivities locally, by considering a single violation independently. This incurs many structural changes, especially if the number of violations is high as in [17].

While the protection methods restrict allowed accesses to RSNs, all the instruments generally must remain accessible by the RSN throughout the design lifecycle. To ensure efficient accessibility, retargeting mechanisms [14], [39], and approaches to minimize the overall test time [40], [41] have been proposed.

III. Modeling

The considered system is shown in Fig. 1, and comprises:
- **DUT**: The design-under-test consists of the functional registers R and the instruments' registers I.
- **RSN**: The RSN accesses the instruments' registers through the scan segments S.

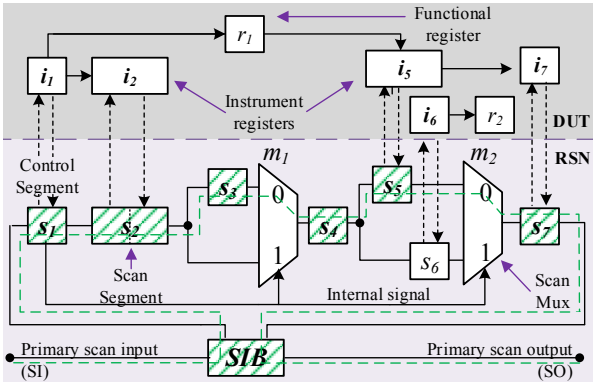

Fig. 1. Considered system: DUT in the upper part, RSN in the lower part with an activated path in green

RSNs consist of the following components:

- The *Scan Segments* $s_j \in S$ are the *scan primitives*, which access the instruments' registers. A scan segment consists of a shift- and an optional shadow register.
- The *Control Signals* cs_i configure a path from a primary scan-in (SI) to a primary scan-out (SO). The internal control signals are driven by the shadow registers of the control scan segments, while the external control signals come from outside of the RSN.
- The *Scan Multiplexers* $m_j \in M$ are the *configuration primitives*, which include certain parts of the RSN into a path, depending on the values of control signals.
- The *Segment Insertion Bits* (SIBs) are the *configuration primitives*, which include or exclude the parts of the RSN into a path, and are modeled as a combination of a scan multiplexer and a scan segment.

The set of scan primitives P includes all the scan segments S and all the scan multiplexers M. A *Scan Configuration* c is defined by the state of the control scan segments. An *Active Scan Path* (ASP) is an acyclic path through selected scan primitives from a primary scan-in to a primary scan-out. In Fig. 1, an initial path starts at the scan-in port, goes towards the SIB, traverses the segments s_1, s_2, s_3, s_4, s_5 and s_7, returns back to the SIB and ends at the scan-out. Computing the control patterns to switch between various active scan paths is called *retargeting*.

The system is modeled as a directed graph G with vertices V and edges E, as shown in Fig. 2 for the RSN from Fig. 1.

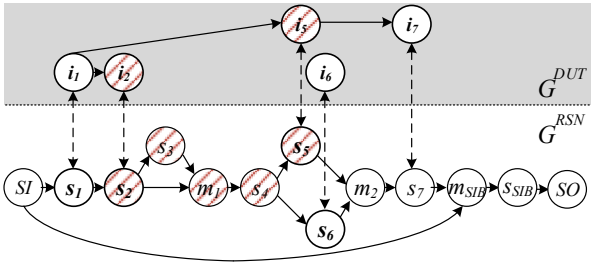

Fig. 2. Combined system graph: DUT graph in the upper part, RSN graph in the lower part

It is constructed of two subgraphs, namely the DUT graph G^{DUT}, the RSN graph G^{RSN}, and the edge set E^{CON}, which represents the connections between the subgraphs. The system graph is defined as follows:

$$V := V^{DUT} \cup V^{RSN} \tag{1}$$

$$E := E^{DUT} \cup E^{RSN} \cup E^{CON} \tag{2}$$

The vertex set V^{DUT} contains all the instruments' registers, which are accessed through the RSN. The edges E^{DUT} represent the direct connectivities between the vertices of the graph or the connectivities through the functional registers.

So, an edge between the instruments i_1 and i_5 in Fig. 2 shows that the instrument i_5 is reachable from the instrument i_1 through the functional register r_1 in the example from Fig. 1.

Each vertex of the RSN graph models either a scan primitive, or corresponds to a primary scan-in or scan-out port. The edges E^{RSN} represent the direct structural connectivities between the vertices of the RSN. The logic signals, which explicitly drive the select-ports of scan primitives, are used for vertex annotation.

IV. SCAR

A. Definitions

Table I contains the major symbols, which are referred to throughout the paper. For each vertex v_j, the set of its direct successors $ds(v_j, G)$ includes all vertices v_k that there exist a directed edge $e_{j,k}$, which connects the vertex v_j to the vertex v_k. The direct predecessors are defined in a similar way.

The set of structural successors $ss(v_j, G)$ of a vertex v_j includes all vertices v_k such that at least one path from v_j to v_k exists in the graph. We say that each successor v_k is structurally reachable from the vertex v_j. The structural predecessors $ss(v_j, G)$ are defined similarly.

Definition 1: A *Functional Path* $path(v_j, v_k)$ between the vertices v_j and v_k is a path, where a valid assignment to the logic signals exists, which selects all the vertices on the path simultaneously.

Definition 2: A *Hybrid Functional Path* is a functional path, which consists of two functional subpaths $path(v_j, v_l)$ and $path(v_l, v_k)$, such that if any two vertices from $\{v_j, v_k, v_l\}$ belong to the RSN graph, the latter one belongs to the DUT graph, and vice versa.

A hybrid path might traverse any combination of subgraphs in the system graph G. In Fig. 2, a hybrid path from the scan segment s_4 to the instrument i_7 can be represented as follows:

$$path(s_4, i_7) := path(s_4, s_5) \& path(s_5, i_5) \,\&\, path(i_5, i_7) \tag{3}$$

A set of functional successors $fs(v_j, G)$ is defined for each vertex v_j, and contains all vertices v_k, which are structurally reachable from v_j, such that either a single functional path from v_j to v_k is sensitizable, or the data from the vertex v_j can be transmitted to the vertex v_k by sequentially activating multiple sensitizable functional paths. The functional predecessors $fp(v_j, G)$ are defined in a similar way.

Definition 3: A *Security Compliance Violation* $viol_x(v_j, v_k)$ is defined as an existence of a connectivity between the vertices v_j and v_k of the RSN graph, which extends the functional connectivity of the original design. The vertex v_j is called the source, and the vertex v_k is the destination of the violation, and x is its index.

For each pair of vertices (v_j, v_k), there might exist at most one violation from v_j to v_k, which considers violating connectivities between these vertices through one or multiple paths.

Example: *In Fig.* 1 *the instrument* i_2 *serves as a storage element for an encryption key, while the instrument* i_5 *contains a wireless communication module. To mitigate information leakage, any data transfer from* i_2 *to* i_5 *is prohibited by a system designer, and a corresponding security property ensures that a corresponding path does not exist in the initial DUT.*

TABLE I. TABLE OF SYMBOLS

For the vertices v_j, v_k in the system graph G, with the control signals $(cs_1, ...cs_n)$:

Symbols	Descriptions
G^{DUT}, G^{RSN}	the DUT and the RSN graph
$dp(v_j, G)$, $ds(v_j, G)$	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
$path(v_j, v_k)$	a functional path from the vertex v_j to the vertex v_k
$viol_x(v_j, v_k)$	a security compliance violation with the id x, the source v_j and the destination v_k
$ESC(v_j, cs_1, ...cs_n)$, $RSC(v_j, v_k, cs_1, ...cs_n)$	the essential and the relative select condition
G_n^{RSN}	a representation of G^{RSN} at an abstraction level n
VS, VT, $OINT$	the sets of sources, destinations, and the optimized set of intermediate vertices

However, due to improper RSN integration, this property is compromised, since an additional path is introduced from the segment s_2 to s_5. This connectivity can be used to build a path between i_2 and i_5 through the RSN, which is shown with a red color in Fig.2, and thereby represents a security violation.

B. General flow

Fig. 3 represents the general flow of the presented security compliance analysis and resynthesis approach, which consists of the following steps:

- **Model construction**: A graph-based model of the system is constructed. It includes the DUT and the RSN.
- **Functional reachability computation**: The functional paths in the RSN are calculated considering the valid assignments to the control signal values. The RSN reachability properties are combined with the security properties of the DUT, and the hybrid paths in the combined system are computed.
- **Security compliance analysis**: The security compliance of a given RSN with the properties of the DUT is analyzed. The connectivities in the RSN, which extend the allowed connectivity of the DUT, are identified as security compliance violations.
- **Resolving 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.
- **Ensuring accessibility**: If certain scan segments are still not accessible after the previous step, some additional connectivities are added, and a modified RSN is constructed.
- **Compliance validation** (shown with a red arrow on the right): 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, as detailed in Section VIII.

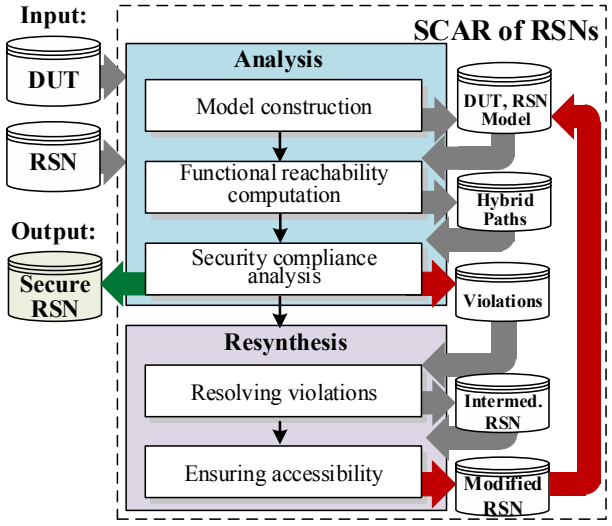

Fig. 3. General flow of the SCAR scheme

V. FUNCTIONAL REACHABILITY AND SECURITY COMPLIANCE ANALYSIS OF RSNS

This section presents an accurate security compliance analysis for RSNs, which considers the properties of the initial design and identifies all the security compliance violations due to the RSN integration. The scheme from [17] is formulated as a graph-based approach and improved in terms of runtime and memory usage. The following steps are performed:

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

A. Security Properties of the DUT

An implicit security specification of the DUT is defined by the reachability properties of the vertices of the initial DUT graph. The structural reachability can be computed by using Floyd-Warshall's algorithm [42]. To identify the functional connectivities inside the DUT, false path analysis [24]–[26] or SAT-based methods [43] can be applied.

The allowed successors and predecessors of the instruments are determined by augmenting the implicit security specification with the explicit specification [23]. 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 DUT graph.

B. 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.

1) Structural Dependencies: The direct successors $ds(v_j, G^{RSN})$ and predecessors $dp(v_j, G^{RSN})$ are obtained for each vertex v_j from the RSN description. Although functional cycles or cyclic ASPs are considered as a bad practice by IEEE Std. 1687 [2], 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. 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.

2) Control Dependencies: Direct data transfer in RSNs is only possible between the scan primitives, which are currently included into an ASP. 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 to determine the connectivities which are functionally activated within a single scan configuration.

Definition 4: The *Essential Select Condition (ESC)* $ESC(v_j, cs_1, ...cs_n)$ for a given scan primitive v_j is a Boolean formula in Conjunctive Normal Form (CNF), which defines the required group of assignments to the control signal values $(cs_1, ...cs_n)$, such that the scan primitive v_j is sensitized.

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 scan primitive, only those control signals are added into its ESC, which are required to place this scan primitive into an active scan path.

If a scan primitive is directly connected to a multiplexer, the ESC demands a specific input of the multiplexer to be selected in order to propagate the data from this scan primitive to the scan-out. This part of the ESC is defined by the *Relative Select Condition (RSC)*. The $RSC(v_j, v_k, cs_1, ...cs_n)$ for a given scan primitive v_j and a scan multiplexer 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 $(cs_1, ...cs_n)$, which are required to drive the address control signal of v_k, in a way that the scan input corresponding to the primitive v_j is selected.

The ESC of each scan primitive 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:

$$ESC(v_j, cs_1, ...cs_n) := \bigvee_{k=1}^{n} [ESC(v_{lk}, cs_1, ...cs_n) \wedge RSC(v_j, v_{lk}, cs_1, ...cs_n)] \quad (4)$$

According to this equation, a given primitive v_j is selected into an active scan path if and only if one of its direct successors is selected ($ESC(v_{lk}, cs_1, ...cs_n)$). If the selected successor is a scan multiplexer, then its scan-in branch, which includes the primitive v_j must be selected by the address control signal ($RSC(v_j, v_{lk}, cs_1, ...cs_n)$).

Fig. 4 shows an example of ESC computation for the scan segment s_2.

3) Valid Scan Dependencies: In RSNs, the data transfer within one CSU-operation from a source scan primitive v_j to a destination scan primitive v_k is possible, only if at least one valid assignment to control signals exists. In this assignment both scan primitives are selected in a valid active scan path, and the scan primitive v_k is reachable from the primitive v_j. If the primitives 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.

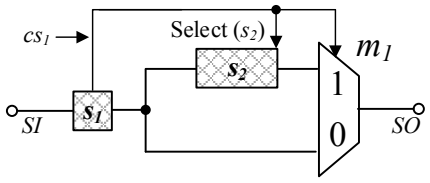

Fig. 4. The signal cs_1 is driven by the shadow register of s_1, and is used as an address control signal of m_1 and to drive the select-port of s_2. Segment s_2 is included into an ASP, if cs_1 equals to one, which means: $ESC(s_2, cs_1) := cs_1$

To verify the condition above, the *Essential Select Conditions* for the structurally connected primitives v_j and v_k are combined by conjunction. The existence of an assignment to the control signals $(cs_1, ...cs_n)$, which satisfies the boolean satisfiability (SAT) instance below, is verified:

$$\begin{aligned} \exists(cs_1, ...cs_n) : [&ESC(v_j, cs_1, ...cs_n) \\ &\wedge ESC(v_k, cs_1, ...cs_n) \\ &\wedge (v_j \in sp(v_k, G^{RSN})] \end{aligned} \tag{5}$$

- If the SAT instance is satisfiable, then an ASP including both primitives can be configured. The satisfying assignments provide the essential values of logic signals to select both primitives 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 primitives are contradicting and an active scan path traversing both primitives does not exist.

4) Functional Reachability Analysis: Data between the instruments can also be propagated through RSNs using multiple sequentially activated paths. The connectivities within single ASPs, which have been computed as shown in Section V-B3, are generalized to determine the functional successors and predecessors of each scan primitive. The number of reconfigurations used to propagate the data between two instruments and thereby the computation efforts in the worst case are limited to the sequential depth of the RSN, which is defined as the length of the longest possible active scan path inside this RSN.

C. Hybrid Connectivity Computation

To compute the connectivity properties after the RSN integration, the hybrid paths traversing both the initial design and the RSN are identified. Therefore, the transitive closure over the system graph is computed by combining the previously determined connectivities within the DUT and the RSN. The presented algorithm to compute the dependencies between the instruments and the scan segments has rather low complexity, and contains two basic steps detailed below:

- **Bridge-dependencies**: The connections between the RSN and the DUT subgraphs are augmented with the connectivities within the subgraphs to build the hybrid partial paths between the instruments and the scan segments (bridges). Following the transitivity property, all the primitives in the RSN, which are reachable from the scan segment reading the data from the instrument, are also reachable from the instrument itself. In this way, all the scan primitives are identified, which are functionally reachable from each given instrument through the RSN. Following the same logic, any instrument reading the data from a particular scan segment is reachable from all the primitives in the RSN, which reach this scan segment.
  Example: *In Fig. 1, the scan segment s_2 is reachable from the instrument i_2. This means that all the scan segments, which are functionally reachable from s_2, are also reachable from i_2 through the RSN.*
- **Instrument connectivities through the RSN**: The partial paths from the first step are combined to find the instrument connectivities through the RSN. Connectivity between the source and the destination instruments exists through the RSN, if at least one intermediate scan primitive exists, which is reachable from the source instrument, and such that the destination instrument is reachable from this scan primitive.
  Example: *In Fig. 1, the scan segment s_6 is reachable from the instrument i_2, and i_6 is reachable from s_6. This implies that i_6 is reachable from i_2 through the RSN.*

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

D. Security Compliance Verification

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

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

Example: *As a result of the security compliance verification, the violating connectivities from the vertex s_1 to s_6; from s_2 to s_5, s_6 and s_7; and finally from s_6 to s_7 are identified.*

VI. Security Preserving Resynthesis

This section presents the automated resynthesis approach.

A. 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 [44], [45].

B. Eliminating the Violating Connectivities

The problem of automated RSN resynthesis is formulated as a minimum cut problem in a multicommodity flow. Since this problem is NP-complete [45], 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. 5):
  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.

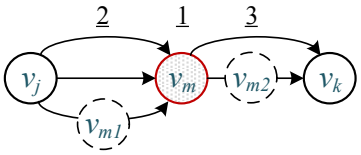

Fig. 5. Node cutting options: a single intermediate vertex v_m is removed to cut all the paths between v_j to v_k.

- The solution is adjusted recursively, as detailed in Section VI-C, 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.
- 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 in Section VI-D below.

C. Divide-and-Conquer Heuristic Overview

1) Base Step: Each violation $viol_x(v_j, v_k)$, is represented the source v_j, the destination v_k, and a temporary edge in between. 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.a for the RSN from Fig. 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.

2) 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 each path between a source and a destination of a violation, 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.b for the RSN from Fig. 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 $weight_L(v_l, G_n^{RSN}, viol_x(v_j, v_k))$ 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.

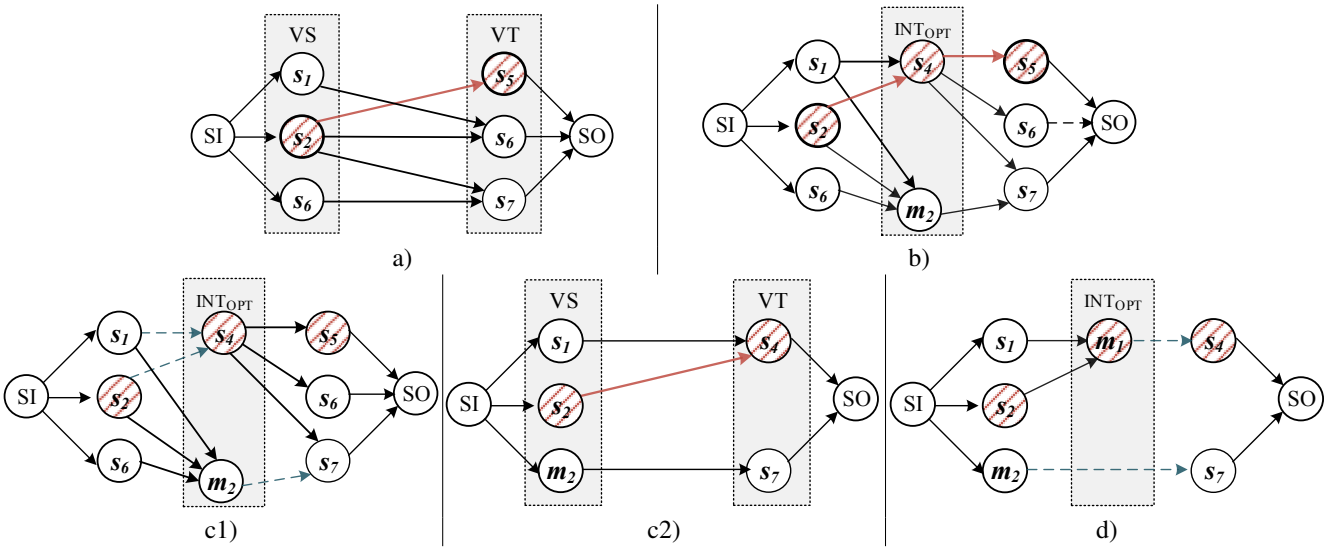

Fig. 6. **a)** *Level-0 graph*: The vertices s_1, s_2, and s_6 serve as sources of violations, while s_5, s_6 and s_7 as destinations. Auxiliary vertices SI and SO serve as a global source and destination. s_6 serves as a source of one violation and as a destination of another violations. A violating path between the vertices s_2 and s_5 from Fig. 2 is shown with a red color.
b) *Level-1 graph construction*: The vertex s_4 has the highest value of a cost function and covers all the violating paths from s_1 to s_6. After s_4 is added into the optimized set, the value of $sel(m_1)$ function is incremented. The same vertex s_4 is automatically selected to cover the paths from s_1 to s_5, and all other violations except the violation from s_6 to s_7. To cover this violation, the vertex m_2 is selected. Finally, the optimized set contains the vertices s_4 and m_2.
c1) *Level-1 graph* is resolved: The violations in the graph are removed by cutting all the paths before s_4 and after m_2, as shown with dashed blue lines.
c2) *Level-2 graph* is constructed: The paths removed in the previous step (c1) are considered as violations.
d) *Level-2 graph* is resolved: The paths after m_1 and after m_2 are removed by cutting the paths from m_1 and s_4, and from m_2 to s_7. The violations, including the one between the vertices s_2 and s_5, are resolved. The computation converges since the solution cannot be further adjusted.

- 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:

$$cost_(v_l, G_n^{RSN}) := \Phi[weight_G(v_l, G_n^{RSN}), \\ weight_L(v_l, G_n^{RSN}, viol_x(v_j, v_k)), \\ mem(v_l, G_n^{RSN})] \tag{6}$$

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:

$$OINT_n := \bigcup_{x=1}^{\#violations} OINT_{n,x} \tag{7}$$

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).

3) 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, as shown in Fig. 7 for the RSN from Fig. 1.

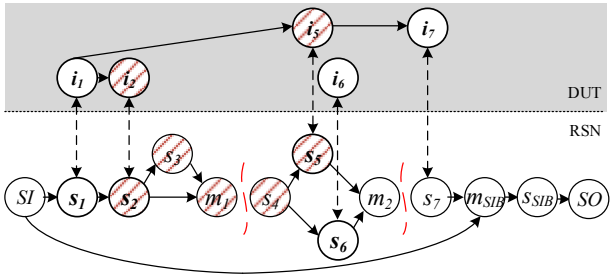

Fig. 7. Secure RSN graph: as decided in Fig. 6.d., the edges from s_3 and s_4 to m_1 and from m_2 to s_7 are removed from the graph to resolve the violations.

D. Ensuring the Accessibility

Since removing a violation corresponds to deleting some edges, 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 the 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 $path_b$ from a primary scan-input to a given segment is further referred to as *backward accessibility*, whereas the existence of a path $path_f$ from a given segment to a primary scan-output is called *forward accessibility*. If the scan segment is forward and backward accessible, and if the activation conditions of the subpaths $path_f$ and $path_b$ are not contradicting, then a path from a scan-in through a given scan segment to a scan-out exists, and the scan segment is accessible.

To ensure the accessibility, first, the *forward accessibility* of the vertices is ensured, as summarized in Algorithm 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*).

For each vertex v_j with no functional successors (*Line 3*), the acccessibility is reintroduced through the following steps:

- *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.

Algorithm 1: Ensuring the forward accessibility

```
1 VertexOrder := Reverse(Order(V^RSN, BFS, SI));
2 for v_j ∈ VertexOrder do
3     if fs(v_j, G^RSN) = ∅ then
4         Find allowed functional successors;
5         Select successor based on optimization criteria;
6         Update the reachability properties;
7     end
8 end
```

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 [14].

The resynthesis is adjustable to the needs of a DfT integrator and various optimization criteria can be applied, e.g.:

- Minimized access latency:

$$minimize[max\ AT(s_i)], s_i \in S, \tag{8}$$

  $AT(s_i)$ denotes the access time of the segment s_i.
  Example: *The maximal access latency among the registers of a periodic BIST, accessed through an RSN online, can be minimized in order to ensure an efficient periodic access, which is compliant with the safety requirements. Using this criteria results in a parallelized RSN, with a higher number of configurable ASPs.*
- Minimized hardware overhead:

$$minimize[\sum_{i=0}^{n} HW(m_i)], m_i \in M, n = |M|, \tag{9}$$

$HW(m_i)$ denotes the hardware overhead of the scan multiplexer m_i.

Example: *The overall hardware overhead, motivated by a number and the complexity of used control primitives, can be minimized in order to reduce the costs of integrating the RSN. The resulting RSN is organized more sequentially with fewer possible ASPs.*

An example for an accessible RSN graph is shown in Fig. 8 for the RSN from Fig.1.

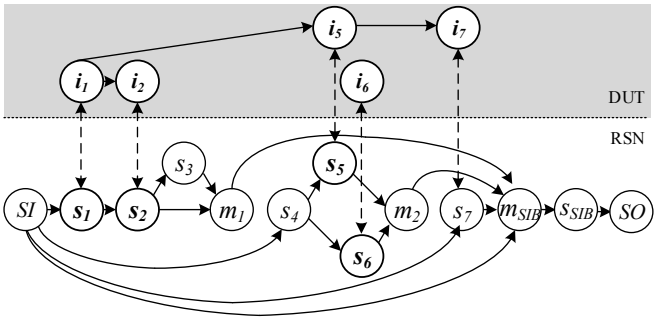

Fig. 8. Accessible RSN graph: the edges from SI, m_1 and m_2 to m_{SIB}, and from SI to s_4 and s_7 are added to ensure the accessibility while preserving the security compliance.

VII. 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 DUT. Iterative validation is used to guarantee that all the violations are resolved, as shown in Fig. 3 with a big red arrow on the right. After applying the resynthesis, the security properties of the resulting RSN are validated once again by using the security compliance analysis, described in Section V:

- If the connectivities in the resulting RSN do not extend the allowed connectivity of the DUT, 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.

VIII. Experimental Results

A. Experimental Setup

The presented SCAR algorithm is implemented in the eda1687 framework as introduced in [14]. All the experiments have been conducted on an Intel(R) Xeon(R) W-2125 CPU at 4.00GHz with 132 GB of main memory. Each evaluated system models the connectivities between the instruments through the DUT, the connectivities inside an RSN, and the ones crossing the boundary between the RSN segments and the accessed instruments. The connectivities between any other components of the DUT can be computed by conventional methods, such as [21-23], and lays out of the scope of this work. To avoid possible bias in the results, we have decided to use the ISCAS'89 [46] benchmarks to represent the connectivities between the instruments through the underlying DUT instead of generating pseudo-random connectivities between the instruments. Each instrument is represented by a flip-flop, and is accessed via a single scan cell of the RSN. The evaluated RSNs have been taken from the well recognized ITC'16 [47] and the DATE'19 [22] benchmark sets, and have a hierarchical structure. The DATE'19 benchmarks come from an industrial partner. They are the largest and also newest benchmarks available to academia. They represent a realistic access mechanism to a memory built-in self-test (BIST) block of a given size. In Table II, the number of scan muxes (Column 2), SIBs (Column 3), scan cells (Column 4) and the highest hierarchy level (Column 5) are given for all the evaluated benchmarks.

TABLE II. Characteristics of benchmarks

Design(1)	#muxes(2)	#sibs(3)	#scan cells(4)	#level(5)
BasicSCB	10	-	176	4
Mingle	13	10	270	3
TreeFlat	24	12	101	2
TreeUnbalanced	28	28	41,887	11
TreeBalanced	46	43	5,581	7
TreeFlat_Ex	60	57	5,194	5
q12710	25	25	26,183	2
a586710	47	-	41,682	3
p34392	142	-	23,261	3
t512505	160	-	77,006	2
p22810	283	283	30,111	3
p93791	653	-	98,637	3
MBIST_1_5_5	15	8	548	4
MBIST_1_5_20	15	8	1,523	4
MBIST_1_20_20	45	23	6,068	4
MBIST_2_5_5	28	16	1,091	4
MBIST_2_5_20	28	16	3,041	4
MBIST_2_20_20	88	46	12,131	4
MBIST_5_5_5	67	40	2,720	4
MBIST_5_20_20	217	115	30,320	4
MBIST_5_100_20	1,017	515	151,520	4
MBIST_5_100_100	1,017	515	671,520	4
MBIST_20_20_20	862	460	121,265	4
MBIST_55_20_5	2,367	1,265	118 970	4
MBIST_100_20_5	8,102	2,300	216,305	4
MBIST_100_100_5	20,102	10,300	1,080,305	4

B. SCAR Runtime Improvement

The SCAR algorithm has been formulated as a graph-based approach. Thanks to the dramatically improved runtime performance, and optimized memory consumption (compared to the matrix-based approach from [17]), larger RSN designs can be analyzed within an acceptable time.

Fig. 9 shows the ratio between the runtime of the presented graph-based scheme and the previously published matrix-based approach 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.

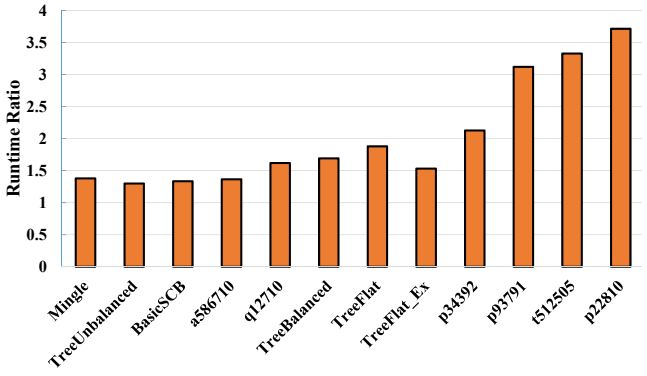

Fig. 9. Runtime ratio between the approach [17] and the presented approach.

For the DATE'19 benchmark set, the runtime performance ratio for the presented approach compared to the matrix-based approach from [17] 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 [17] 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 [18] is significantly larger in comparison to the ILP equations required in the proposed method. Therefore, for all other benchmarks from the DATE'19 benchmark set, the computation is performed with the presented improved approach only.

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.

C. DUT-RSN Reachability Dependencies

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 in the DUT 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. 10 shows the dependency between the fraction of prohibited connectivities in the DUT and the average fraction of violations in the RSN.

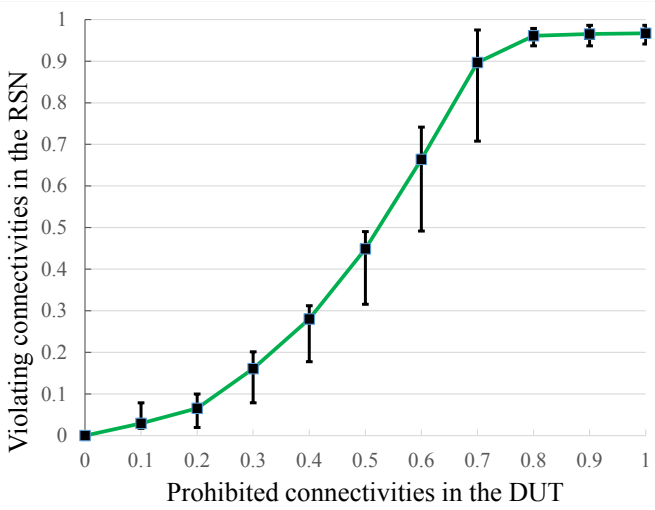

Fig. 10. The reachability properties of the DUT and the RSN.

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

D. Total Flow Check

The flow from Fig. 3 is performed to securely integrate an RSN into a given DUT. 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 experimental results are shown in Table III:

- 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.
- Thirdly, all the violations have been resolved with a few iterations of the flow from Fig. 3 (Column 6, #$iter$), by removing just a few edges (Column 7, #$removed$).

- 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, as defined in Section VI-D. 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 security compliant RSNs are normalized with respect to the values obtained for the initial RSNs and are shown in Columns 8, 9, 10, 11.

- 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 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.

Compared to generating fully parallel RSNs, using the presented method is beneficial. In a fully parallel RSN, all the scan cells will be located on different branches of a scan multiplexer. Therefore, the number of required changes for each initial benchmark would be equal to the number of scan cells provided in Column 4 of Table II, which is much higher than the changes required by the presented method (Column 7 in Table III). Maximum latency in terms of shift cycles will be reduced, since the lengths of scan paths will be decreased. However, to read and/or write data to multiple instruments through such an RSN, additional capture and update cycles are required, which increases overall access time significantly if the number of parallel branches is too high. Hardware overhead in a fully parallel RSN, according to Eq.9, is calculated as the hardware overhead of the scan multiplexer, which is defined as the number of cascading two-input scan multiplexers required to access all the parallel branches. Therefore, for a fully parallel RSN the hardware overhead is much higher than the one obtained by the presented method (Columns 9 and 11 in Table III).

IX. Conclusion

This paper presents a complete approach for security preserving integration of RSNs, which ensures the compliance of the resulting RSN with the requirements of the DUT. The presented approach overcomes the high sequential depth of RSNs, and accurately analyzes the RSN functional dependencies. It considers hybrid paths through the DUT and the RSN, as well as the retargeting capabilities of RSNs, and identifies all the security violations due to the RSN integration. Based on the results of a security compliance analysis, the presented resynthesis approach applies a minimized number of structural changes to the RSN in order to resolve all the identified violations.

An efficient divide-and-conquer heuristic is presented, which avoids exponential complexity in average case. The automated resynthesis is flexible, and allows to specify additional optimization criteria based on the DfT integrator needs. The feasibility of the presented method, as shown in the experimental results, allows to securely integrate an RSN into the DUT with an acceptable runtime and a minor hardware overhead.

Acknowledgment

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.

References

[1] "IEEE Standard for Test Access Port and Boundary-Scan Architecture," *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)*, pp. 1–444, May 2013.

[2] "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," *IEEE Std 1687-2014*, pp. 1–283, Dec. 2014.

[3] 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)*, 2005, pp. 1 – 8.

[4] 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)*, Virtual, June 2021, pp. 1–7.

[5] J. D. Rolt, A. Das, G. D. Natale, M. Flottes, B. Rouzeyre, and I. Verbauwhede, "Test Versus Security: Past and Present," *IEEE Trans. on Emerging Topics in Computing*, vol. 2, no. 1, pp. 50–62, Mar. 2014.

[6] Free60 SMC Hack. Available https://free60project.github.io/wiki/SMC_Hack/. Accessed December 09, 2021 [Online].

[7] 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].

[8] 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)*, vol. 40, no. 7, pp. 1301–1314, 2021.

[9] S. Kan and J. Dworak, "IJTAG Integrity Checking with Chained Hashing," in *Proc. IEEE Int'l Test Conf. (ITC)*, Oct. 2018, pp. 1–10.

[10] 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)*, May 2018, pp. 1–6.

[11] 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.

[12] 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)*, vol. 25, no. 10, pp. 2287–2293, Oct. 2006.

[13] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, "Securing Designs against Scan-Based Side-Channel Attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 325–336, Oct 2007.

[14] 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)*, vol. 20, no. 2, pp. 30:1–30:27, 2015.

TABLE III. SECURITY COMPLIANT RSN INTEGRATION

	Security Compliance Analysis				Resynthesis					
					Resolve		Reduce Latency		Reduce Overhead	
(1) Design	(2) $\#struct.$	(3) $\#ASP$	(4) $\#func.$	(5) $\#viol.$	(6)$\#iter$	(7) $\#removed$	(8) $latency$	(9) $overhead$	(10) $latency$	(11) $overhead$
BasicSCB	181	175	178	45	1	8	0.91	1.09	1.01	0.93
Mingle	230	155	205	83	1	9	0.90	1.02	0.97	0.93
TreeFlat	300	263	300	103	1	9	0.90	1.05	0.98	0.93
TreeUnbalanced	2,016	1,820	1,987	340	1	9	0.99	0.96	0.99	1.01
TreeBalanced	4,272	2,688	2,899	822	1	18	0.98	1.02	0.97	0.94
TreeFlat_Ex	7,869	6,987	7,116	1450	1	20	0.76	0.97	1.15	0.85
q12710	1,275	1,020	1,219	614	1	22	0.72	1.21	1.13	0.80
a586710	1,430	1,034	1,599	345	2	37	0.63	1.67	1.12	0.97
p34392	15,937	12,122	14,435	2,187	3	21	0.75	1.49	1.12	0.96
t512505	41,328	30,623	37,677	1,346	2	36	0.66	1.61	1.16	0.93
p22810	137,550	97,731	125,637	2,104	1	127	0.78	1.60	1.10	1.01
p93791	721,269	523,454	652,825	18,610	3	463	0.59	1.72	1.18	0.85
MBIST_1_5_5	30,193	29,600	29,605	967	1	101	0.18	4.00	1.34	1.00
MBIST_1_5_20	231,043	222,124	229,475	365	1	52	0.28	3.24	1.56	1.01
MBIST_1_20_20	929,218	916,123	923,015	356	1	48	0.24	3.33	1.12	0.97
MBIST_2_5_5	60,327	57,122	59,153	1,056	1	104	0.29	2.12	1.12	0.98
MBIST_2_5_20	462,027	422,173	458,903	5,591	2	187	0.31	2.21	1.15	1.10
MBIST_2_20_20	1,858,377	1,804,592	1,845,983	3,437	2	358	0.49	1.99	1.15	1.02
MBIST_5_5_5	150,783	142,861	147,866	23,177	1	4,826	0.22	3.55	1.56	1.01
MBIST_5_20_20	4,645,908	4,124,739	4,614,941	63,284	1	6,712	0.19	4.05	1.23	0.97
MBIST_5_100_20	23,947,908	22,112,834	23,793,341	56,452	1	2,561	0.45	2.13	1.12	0.99
MBIST_5_100_100	452,167,908	423,175,253	451,493,341	54,765	1	972	0.65	1.16	1.18	1.09
MBIST_20_20_20	18,584,778	15,267,365	18460946	23,683	1	1,835	0.49	1.67	1.42	0.99
MBIST_55_20_5	6,929,683	6,605,819	6,803,666	17,349	2	1,240	0.69	1.46	1.14	0.98
MBIST_100_20_5	12,619.618	11,859.933	12.390,506	87,275	2	5,784	0.34	2.60	1.37	0.96
MBIST_100_100_5	77,299,618	75,738,342	76,158,506	154,235	3	6,246	0.35	2.56	1.14	0.98

[15] A. Tsertov, A. Jutman, K. Shibin, and S. Devadze, "IEEE 1687 Compliant Ecosystem for Embedded Instrumentation Access and In-Field Health Monitoring," in *In Proc. IEEE AUTOTESTCON*, Sep. 2018, pp. 1–9.

[16] A. M. Y. Ibrahim and H. G. Kerkhoff, "DARS: An EDA Framework for Reliability and Functional Safety Management of System-on-Chips," in *Proc. IEEE Int'l Test Conf. (ITC)*, Nov. 2019, pp. 1–10.

[17] 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)*, Nov. 2019, pp. 1–9.

[18] 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)*, Nov. 2020, pp. 1–10.

[19] 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)*, May 2016, pp. 1–6.

[20] P. Raiola, T. Paxian, and B. Becker, "Minimal Witnesses for Security Weaknesses in Reconfigurable Scan Networks," in *Proc. IEEE European Test Symp. (ETS)*, May 2020, pp. 1–6.

[21] 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)*, Jul. 2018, pp. 91–96.

[22] 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)*, Mar. 2019, pp. 1–6.

[23] M. A. Kochte, M. Sauer, L. R. Gomez, P. Raiola, B. Becker, and H. Wunderlich, "Specification and Verification of Security in Reconfigurable Scan Networks," in *Proc. IEEE European Test Symp. (ETS)*, May 2017, pp. 1–6.

[24] Z. Hanna and V. M. Purri. (2013, Apr.) Verifying Security Aspects of SoC Designs with Jasper App. [Online]. Available: https://www.edn.com/

[25] 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)*, Nov. 1998, pp. 392–395.

[26] 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)*, Oct. 2017, pp. 1–10.

[27] 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*, vol. 38, no. 3, pp. 538–550, 2019.

[28] 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)*, Jul. 2013, pp. 228–229.

[29] X. Li, W. Li, J. Ye, H. Li, and Y. Hu, "Scan chain based attacks and countermeasures: A survey," *IEEE Access*, vol. 7, pp. 85 055–85 065, 2019.

[30] H. Liu and V. D. Agrawal, "Securing IEEE 1687-2014 Standard Instrumentation Access by LFSR Key," in *Proc. IEEE Asian Test Symp. (ATS)*, Nov. 2015, pp. 91–96.

[31] 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)*, May 2019, pp. 1–6.

[32] 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. Conf. on Design, Automation Test in Europe (DATE)*, Mar. 2014, pp. 1–6.

[33] 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)*, Nov. 2017, pp. 1–10.

[34] 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)*, vol. 34, no. 6, pp. 937–946, June 2015.

[35] M. Portolan, V. Reynaud, P. Maistri, and R. Leveugle, "Dynamic Authentication-Based Secure Access to Test Infrastructure," in *Proc. IEEE European Test Symp. (ETS)*, May 2020, pp. 1–6.

[36] 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)*, 2019, pp. 1–6.

[37] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Access Port Protection for Reconfigurable Scan Networks," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 30, no. 6, pp. 711–723, 2014.

[38] A. Atteya, M. A. Kochte, M. Sauer, P. Raiola, B. Becker, and H.-J. Wunderlich, "Online Prevention of Security Violations in Reconfig-

urable Scan Networks," in *Proc. IEEE European Test Symp. (ETS)*, May 2018, pp. 1–6.

[39] A. 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)*, May 2016, pp. 1–6.

[40] 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)*, Apr. 2018, pp. 129–134.

[41] 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)*, pp. 1–1, Apr. 2020.

[42] S. Warshall, "A Theorem on Boolean Matrices," *Journal of the ACM (JACM)*, vol. 9, no. 1, pp. 11–12, Jan. 1962.

[43] M. Soeken, P. Raiola, B. Sterin, B. Becker, G. De Micheli, and M. Sauer, *Proc. 12th Int'l Haifa Verification Conference (HVC)*. Springer, 2016, ch. SAT-Based Combinational and Sequential Dependency Computation, pp. 1–17.

[44] L. R. Ford and D. R. Fulkerson, "A Suggested Computation for Maximal Multi-Commodity Network Flows," *Management Science*, vol. 5, pp. 97–101, 1958.

[45] A. Hall, S. Hippler, and M. Skutella, "Multicommodity Flows Over Time: Efficient Algorithms and Complexity," *Theoretical Computer Science*, vol. 379, no. 3, pp. 387 – 404, 2007.

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

[47] 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)*, Nov. 2016, pp. 1–10.

Natalia Lylina (S'17) 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. Since 2017 she is with the Institute of Computer Architecture and Computer Engineering at the University of Stuttgart as a PhD student. Her research interests include dependable systems, test and diagnosis infrastructure and reconfigurable scan networks.

Chih-Hao Wang (S'17-M'21) 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. His research interests include VLSI testability and reliability, concurrent error detection, and reconfigurable scan networks.

Hans-Joachim Wunderlich (M'85–LF'20) 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 he was the director of the Institute of Computer Architecture and Computer Engineering at the University of Stuttgart, Germany.

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.