Detecting and Resolving Security Violations in Reconfigurable Scan Networks

Raiola, Pascal; Kochte, Michael A.; Atteya, Ahmed; Rodríguez Gómez, Laura; Wunderlich, Hans-Joachim; Becker, Bernd; Sauer, Matthias

Proceedings of the 24th IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS'18), Platja d'Aro, Spain, 2-4 July 2018, pp. 91-96

doi: http://dx.doi.org/10.1109/IOLTS.2018.8474188

Abstract: Reconfigurable Scan Networks (RSNs) allow flexible access to embedded instruments for post-silicon validation and debug or diagnosis. However, this scan infrastructure can also be exploited to leak or corrupt critical information as observation and controllability of registers deep inside the circuit are increased. Securing an RSN is mandatory for maintaining safe and secure circuit operations but difficult due to its complex data flow dependencies. This work proposes a method that detects security violations and transforms a given insecure RSN into a secure RSN for which the secure data flow as specified by a user is guaranteed by construction. The presented method is guided by user-defined cost functions that target e.g. test performance or wiring cost. We provide a case study and experimental results demonstrating the applicability of the method to large designs with low runtime.

Preprint

Detecting and Resolving Security Violations in Reconfigurable Scan Networks

Pascal Raiola*, Michael A. Kochte‡, Ahmed Atteya‡, Laura Rodríguez Gómez‡,
Hans-Joachim Wunderlich‡, Bernd Becker*, Matthias Sauer*
*University of Freiburg, Germany
‡ITI, University of Stuttgart, Germany

***Abstract*—Reconfigurable Scan Networks (RSNs) allow flexible access to embedded instruments for post-silicon validation and debug or diagnosis. However, this scan infrastructure can also be exploited to leak or corrupt critical information as observation and controllability of registers deep inside the circuit are increased.**

Securing an RSN is mandatory for maintaining safe and secure circuit operations but difficult due to its complex data flow dependencies.

This work proposes a method that detects security violations and transforms a given insecure RSN into a secure RSN for which the secure data flow as specified by a user is guaranteed by construction. The presented method is guided by user-defined cost functions that target e.g. test performance or wiring cost. We provide a case study and experimental results demonstrating the applicability of the method to large designs with low runtime.

***Keywords*—Reconfigurable Scan Network, Hardware Security, Data Flow, IEEE Std 1687**

I. Introduction

Today's complex circuitry employs a great variety of on-chip functional and non-functional instrumentation to facilitate e.g. on-chip diagnosis, post-silicon validation and bring-up, therefore allowing test/debug access to the circuit [1]. In order to cope with the complexity of connecting these instruments, reconfigurable scan networks *(RSNs)* as standardized by e.g. IEEE Std 1149.1-2013 or IEEE Std 1687-2014 are increasingly deployed in (industrial) designs. Such RSNs allow configuration and changes to the active scan path and hence provide flexible and scalable access to embedded instruments.

However, due to the flexibility and powerful observability and controllability properties of an RSN, security properties might be compromised. They need to be guaranteed in order to allow secure operations and prevent an attacker to use the infrastructure as a side channel to leak sensitive information [2], [3].

In this work, we present a method that detects security violations in a given RSN and structurally transforms a given (insecure) RSN into a *secure* RSN based on a user-given security specification [4]. The security specification in our model is provided by defining a trust category for each segment of an RSN to model its trustworthiness. Furthermore, for each segment a set of accepted trust categories to model its data sensitivity is given. Based on these user-specified information, paths that may leak or corrupt sensitive data by passing segments, which do not guarantee an adequate trustworthiness, are prohibited. Such insecure paths are identified by an efficient traversal algorithm over the RSN structure. In a second step, we gradually change the structure as well as the connectivity of the given RSN in order to structurally prevent every insecure scan path while still maintaining accessibility of the individual RSN segments. The proposed method is guided by a cost heuristic, that optimizes the resulting RSN with respect to different cost criteria like the overall access latency and thereby tries to keep the modifications as small as possible.

The applicability of the presented method is demonstrated by an experimental evaluation using standard RSN benchmarks. As we will demonstrate, the computational runtime is in the order of a few seconds for all tested benchmarks. Additionally, we give example heuristics to guide the search toward different goals and report an analysis of these different heuristics.

The remainder of the paper is organized as follows: Section II summarizes the related work on securing RSNs. Section III gives the required background on the underlying model. The method to transform an insecure RSN into a secure RSN is explained in greater detail in Section IV. Section V provides the experimental results. Section VI concludes the paper.

II. Related Work

The potential conflict between testability and security has been already addressed for conventional design-for-test infrastructures such as standard scan chains, e.g. in [5]–[7]. The data flow in conventional scan infrastructure is always static and thereby much less complex than in RSNs.

Security of reconfigurable scan infrastructure has been tackled in various works, with diverse approaches to defend against attacks. Approaches using authorization techniques were e.g. presented by the authors of [8], who proposed an authorization instrument for access management as well as by the authors of [9], [10], who introduced a technique where scan segments are skipped, unless a secret key is provided. Secure test wrappers are introduced e.g. by [11] and [12], where the technique from [11] functions without any hard-coded secrets in the design and in [12] flip-flops are reused to achieve little area overhead.

Various techniques to obfuscate the scan data have been introduced: E.g. inverters get inserted at unknown positions [13], the scan chain is partitioned into sub-chains and the access to the sub-chains is pseudo-randomized [14], obfuscation is achieved with the use of state-dependent flip-flops [15] and XOR-gate confusion is added [16]. The authors of [17] present a compaction approach, where due to on-chip test comparison the full test response is compacted to one bit without a loss in test quality or negative impact on diagnostic of modeled faults.

A sole alteration of sensitive data in a maybe even non-deterministic way might cause security issues. Therefore in this work the scan infrastructure must not allow any *insecure* data traversal over a scan path.

In [18], [19] a filter is introduced locally at the interface of the RSN (TAP), that only allows a precomputed set of secure scan-in access sequences. [20] introduces a filter, that monitors the security requirements online and can prevent forbidden accesses.

Security violations might occur for registers which are inseparable by scan path configuration, forcing a filter to make every such register pair inaccessible. In contrast the method presented in this paper re-designs the RSN, to include all scan registers in the scan infrastructure.

The use of e-fuses or a wafer saw to fully deactivate the scan path has been proposed e.g. in [21] and [22], making any in-field use of the scan infrastructure (e.g. built-in self-test) impossible and is therefore not considered in this work.

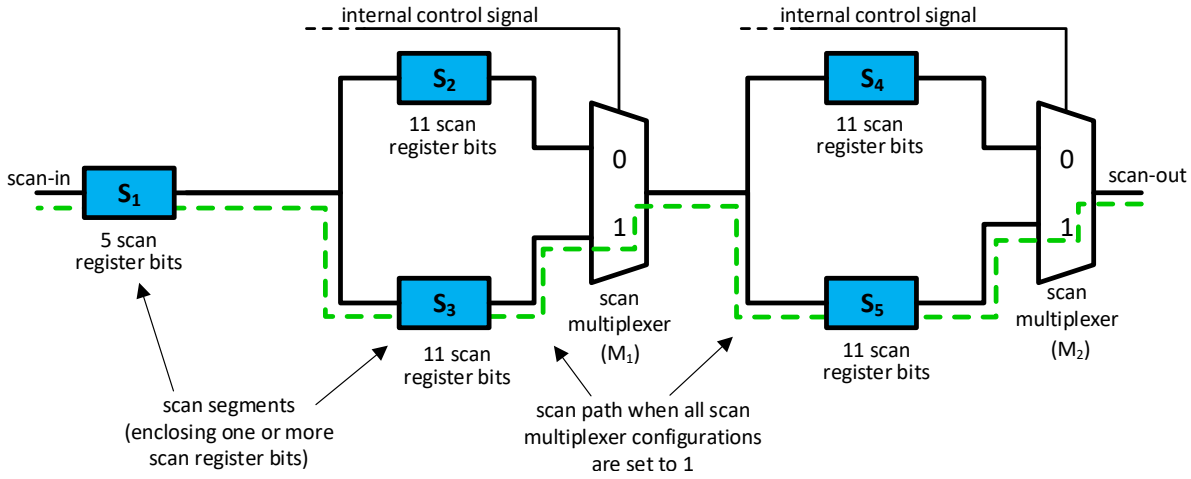

Fig. 1: Reconfigurable scan network with five scan segments (S_1 to S_5)

A formal description of data dependency in sequential circuits is introduced in [23], but did not consider any security aspects. We adapt the model introduced in [4] to specify access permissions and restrictions of an RSN.

To the best of our knowledge, the challenge of guaranteeing secure data flow in the design of reconfigurable scan infrastructure without the need to add complex control circuitry, yet still maintaining flexible in-field access, is only considered in this work.

III. Underlying Model

A. Reconfigurable Scan Network: Running Example

We focus on reconfigurable scan networks (RSNs) as standardized by e.g. IEEE Std 1149.1-2013 or IEEE Std 1687-2014. Figure 1 shows an example RSN consisting of five scan segments (S_1 to S_5) and two scan multiplexers M_1 and M_2. Note that the scan segments contain a potentially large amount of scan register bits (5 and 11 respectively, in the running example RSN of Fig. 1).

In an RSN multiple active scan paths are configurable, the example of Figure 1 pictures the active scan path if both scan multiplexers are set to 1 (dashed line).

An RSN is equipped with three global signals *capture*, *shift* and *update*, which are controlled by the TAP controller and are exclusively active, enabling either the *capture-*, *shift-* or *update-phase*, respectively. In the capture phase, the scan registers are loaded with data from the underlying circuit, while during the shift phase, data is shifted through the segments' scan registers. In the update phase, data from the scan registers is loaded into the optional shadow registers.

B. Security Threats

In this work, two security threats are investigated as well as a countermeasure against both threats.

1) Attack via untrusted Third-party IP: Scan infrastructure is in general deeply weaved into the circuit, connecting instruments from different sources with each other by a common interface. Hence, shifting confidential data over a scan path involving an untrusted third-party module needs to be ruled out. Third party modules often need to be considered as untrusted, as their trustworthiness cannot be guaranteed.

In other words the security threat described above has to be taken into account, if the following holds:

- One scan segment S contains *sensitive data*.
- There is a scan path through S and an *untrusted* module.

In Chapter III-C, we will present a formal definition of *data sensitivity* and *trustworthiness* to allow the modeling of such threats.

2) Using Scan Path Branches for a Side-Channel Attack: The scan path in Figure 1 (dashed line) shows the data flow from the scan-in port over the scan segments to the scan-out port. The data stream not only follows this path, but additionally branches at each fan-out (cf. Figure 2, blue dashed lines). If S_2 or S_4 are inside a third-party module, it should be considered, that data on the scan path also enters the respective third-party module.

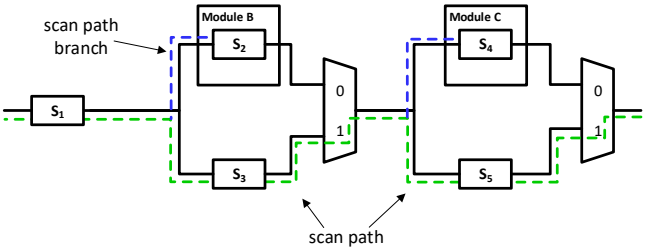

Fig. 2: Modules adjacent to the scan path access scan data via scan path branches.

Therefore even though a module is not part of the current scan path, it might still leak sensitive data e.g. via a side-channel attack, which is also taken into account by our security model.

3) Proposed Method: This work investigates the RSN for prohibited connections between scan segments and uses rewiring to structurally eliminate every such prohibited connection. It therefore serves as countermeasure for the security threats of both Chapter III-B1 and Chapter III-B2.

C. Security Specification

As already mentioned, we adapt the model introduced in [4] to specify access permissions and restrictions to instruments attached to scan segments in an RSN. Each scan segment S is denoted with a *trust category* $T(S)$, characterizing the trustworthiness of the segment (or its surrounding core). Furthermore, for each segment a set of accepted trust categories $AT(S)$ is given. $AT(S)$ serves as a measure of data sensitivity, i.e. the category of secrecy or required protection of the data stored in or read from S. To fulfill data sensitivity issues, any active scan path through S must only traverse registers with a trust category in $AT(S)$. This allows for flexible specifications, where the trust categories can be e.g. linearly ordered or not ordered at all.

The assignment of all segments to their respective trust categories and confidentialities is referred to as *(data flow)*

security specification. The security specification is violated if there exists an active scan path over two segments $\mathtt{S_x}$ and $\mathtt{S_y}$, where the trust category of one segment is not contained in the set of accepted trust categories of the other segment:

$$T(\mathtt{S_x}) \notin AT(\mathtt{S_y}) \quad \text{or} \quad T(\mathtt{S_y}) \notin AT(\mathtt{S_x}) \tag{1}$$

In case of $T(\mathtt{S_x}) \notin AT(\mathtt{S_y})$ the low trustworthiness of $\mathtt{S_x}$ could be used to access confidential data mirrored by the scan segment $\mathtt{S_y}$ and vice-versa for $T(\mathtt{S_y}) \notin AT(\mathtt{S_x})$.

If an RSN comprises no such violation, it is called *(data flow) secure*. The data flow security specification of the running example RSN (cf. Figure 1) is shown in Table I:

Scan Segment(s)	Trust (T)	Accepted Trust (AT)
$\mathtt{S_1}$, $\mathtt{S_3}$, $\mathtt{S_4}$	a	$\{a, b, c\}$
$\mathtt{S_2}$	b	$\{a, b\}$
$\mathtt{S_5}$	c	$\{a, c\}$

TABLE I: Security specification of the example RSN

Note that trust category a (of segments $\mathtt{S_1}$, $\mathtt{S_3}$, $\mathtt{S_4}$) is contained in every set of accepted trust categories (last column). Consequently, data from and toward every scan segment is allowed to be shifted through scan segments of trust category a; thus, segments of trust category a, are accepted as trustworthy by each segment/module. In contrast, scan segments $\mathtt{S_2}$ and $\mathtt{S_5}$ are e.g. placed inside different third party IP modules and are hence annotated with trust categories, that are not universally accepted as trustworthy.

It can also be seen that data of scan segments with accepted trust set $\{a, b, c\}$ can be shifted through scan segments of any trust category, as every trust category is contained in the set. Therefore such scan segments do not mirror data, which is too confidential for other modules. Especially for large RSNs, it is in general not straightforward to state whether an RSN is secure.

Note that the given RSN is not secure, as segments $\mathtt{S_2}$ and $\mathtt{S_5}$ may exchange data, which violates the given security specification. In general, efficiently detecting such security violations is non-trivial as it requires an in-depth analysis of the RSN structure and its security specification, as presented in the following Chapter IV-A.

IV. Detection and Correction

An overview of the proposed method is provided in Figure 3. At first a structural analysis is performed to find security violations in the existing RSN structure (cf. Subchapter IV-A). A security violation – if existent – is selected and multiple change candidates to resolve the security violation are investigated (cf. Subchapter IV-B), utilizing a cost function, that aims at specific design goals (cf. Subchapter IV-C). The change candidate that led to the lowest cost will be applied.

As long as a security violation is found, the above described steps for the corresponding insecure path are executed.

A. Analysis for Security Violations

This work uses a structural analysis of the RSN for an efficient check of the security specification. To do so, we annotate each scan segment $\mathtt{S}$ additionally with the *unified trust* $\overrightarrow{UT}$ and the *common accepted trust* $\overrightarrow{CAT}$:

- $\overrightarrow{UT(\mathtt{S})}$ is the union of the trust category of $\mathtt{S}$ with the trust categories of all predecessors.
- $\overrightarrow{CAT(\mathtt{S})}$ describes the intersection of $AT(\mathtt{S})$ (the set of *accepted trust categories of* $\mathtt{S}$*) with the AT-sets of all predecessors of* $\mathtt{S}$.

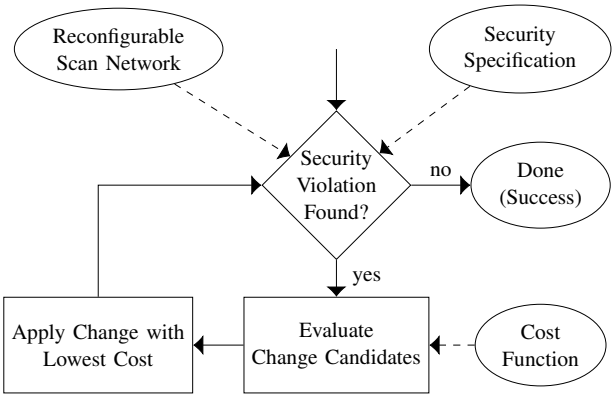

Fig. 3: Flow of proposed method.

The respective values of all scan segments of the example RSN (cf. Figure 1) are given in Table II:

Scan Segment(s)	$\overrightarrow{UT}$	$\overrightarrow{CAT}$
$\mathtt{S_1}$	$\{a\}$	$\{a, b, c\}$
$\mathtt{S_2}$	$\{a, b\}$	$\{a, b\}$
$\mathtt{S_3}$	$\{a\}$	$\{a, b, c\}$
$\mathtt{S_4}$	$\{a, b\}$	$\{a, b\}$
$\mathtt{S_5}$	$\{a, b, c\}$	$\{a\}$

TABLE II: Derived security values of the example RSN

Both $\overrightarrow{UT}$ and $\overrightarrow{CAT}$ can be iteratively calculated in a preprocessing step, by traversing through the the RSN in topological order. In the example RSN, the values for the scan segment $\mathtt{S_5}$ were calculated as follows:

$$\overrightarrow{UT(\mathtt{S_5})} = T(\mathtt{S_5}) \cup \overrightarrow{UT(\mathtt{S_2})} \cup \overrightarrow{UT(\mathtt{S_3})} \tag{2}$$

$$\overrightarrow{CAT(\mathtt{S_5})} = AT(\mathtt{S_5}) \cap \overrightarrow{CAT(\mathtt{S_2})} \cap \overrightarrow{CAT(\mathtt{S_3})} \tag{3}$$

After this preprocessing, (data flow) security violations can be locally detected for each scan segment: The target is to find a violation, i.e. an insecure path from a (predecessor) scan segment $\mathtt{S}_{Pr}$ to a (successor) scan segment $\mathtt{S}_{Su}$. Thus, as displayed in Formula 1, either the trust category of $\mathtt{S}_{Pr}$ conflicts with $\mathtt{S}_{Su}$'s accepted trust – $T(\mathtt{S}_{Pr}) \notin AT(\mathtt{S}_{Su})$ – or the trust category of $\mathtt{S}_{Su}$ conflicts with $\mathtt{S}_{Pr}$'s accepted trust – $T(\mathtt{S}_{Su}) \notin AT(\mathtt{S}_{Pr})$.

- Instead of checking $T(\mathtt{S}_{Pr}) \notin AT(\mathtt{S}_{Su})$ for each pair of scan segments, it should be noted, that the union $\overrightarrow{UT(\mathtt{S}_{Su})}$ contains $T(\mathtt{S}_{Pr})$. Therefore it is possible to instead *locally* check

$$\overrightarrow{UT(\mathtt{S}_{Su})} \notin AT(\mathtt{S}_{Su}) \tag{4}$$

  for each scan segment $\mathtt{S}_{Su}$.
- Similarly, if $T(\mathtt{S}_{Su}) \notin AT(\mathtt{S}_{Pr})$, then $T(\mathtt{S}_{Su})$ is also not contained in the intersection $\overrightarrow{CAT(\mathtt{S}_{Su})}$. Thus, instead of checking $T(\mathtt{S}_{Su}) \notin AT(\mathtt{S}_{Pr})$ for each scan segment pair, the following property can be *locally* checked:

$$T(\mathtt{S}_{Su}) \notin \overrightarrow{CAT(\mathtt{S}_{Su})} \tag{5}$$

The possibility to locally test for security violations of potentially distant scan segments is summarized in the following lemma:

Lemma 1: In every security violation is one scan segment $\mathtt{S}$ involved, for which $\overrightarrow{UT(\mathtt{S})} \notin AT(\mathtt{S})$ or $T(\mathtt{S}) \notin \overrightarrow{CAT(\mathtt{S})}$ holds.

If by utilizing Lemma 1, a security violation is found for a scan segment, a backward traversal identifies a violating predecessor over an insecure path. In Tables I and II it can be seen, that $\overrightarrow{UT(\mathtt{S}_5)} \notin AT(\mathtt{S}_5)$ as well as $T(\mathtt{S}_5) \notin \overrightarrow{CAT(\mathtt{S}_5)}$ holds, which both imply a security violation. By investigating one of these implications further, a backward traversal leads to the predecessor $\mathtt{S}_2$. Hence, it must be prohibited, to shift data from $\mathtt{S}_2$ into $\mathtt{S}_5$. Otherwise the low trustworthiness of one scan segment could be used to access confidential data mirrored by the other scan segment.

B. Candidates to Resolve the Security Violation

Once a security violation has been identified, we modify the RSN structure in a way that the insecure data path does no longer exist. In order to do so, multiple resolve options are investigated for every pair of *directly* connected ($\mapsto$) segments on the insecure scan path.

In the example RSN (cf. Figure 1), $\mathtt{S}_2$ and $\mathtt{S}_5$ were found to cause a security violation over the path $\mathtt{S}_2 \mapsto \mathtt{M}_1 \mapsto \mathtt{S}_5$. The resolve options are then investigated once for a cut at $\mathtt{S}_2 \mapsto \mathtt{M}_1$ and once for a cut at $\mathtt{M}_1 \mapsto \mathtt{S}_5$.

In general, for a pair of *directly* connected segments $\mathtt{S}_x$ and $\mathtt{S}_y$, at first the connection $\mathtt{S}_x \mapsto \mathtt{S}_y$ is cut. Then, to prevent unconnected scan segments, the method structurally searches the RSN to find potential new successor segments of $\mathtt{S}_x$ (called $Succ_{pot}(\mathtt{S}_x)$) and potential new predecessor segments of $\mathtt{S}_y$ (called $Pred_{pot}(\mathtt{S}_y)$), respectively.

The search for $Succ_{pot}(\mathtt{S}_x)$ starts at $\mathtt{S}_x$ and branches, whenever a fan-out is traversed. Scan segments that cause a security violation with $\mathtt{S}_x$ are skipped. A search branch terminates as soon as a scan segment is reached, that does not cause a security violation with $\mathtt{S}_x$. If a search branch reaches the scan-out port, the scan-out is added to $Succ_{pot}(\mathtt{S}_x)$. Thus $Succ_{pot}(\mathtt{S}_x)$ is not empty, as each search branch reaches a segment that will be contained in $Succ_{pot}(\mathtt{S}_x)$.

$Pred_{pot}(\mathtt{S}_y)$ is computed analogously but branches at every multiplexer instead of every fan-out and proceeds in the direction of the scan-in port.

In the example RSN (cf. Figure 1), the above described search was executed once for a cut at $\mathtt{S}_2 \mapsto \mathtt{M}_1$ and once for a cut at $\mathtt{M}_1 \mapsto \mathtt{S}_5$. The latter is displayed in Figure 4:

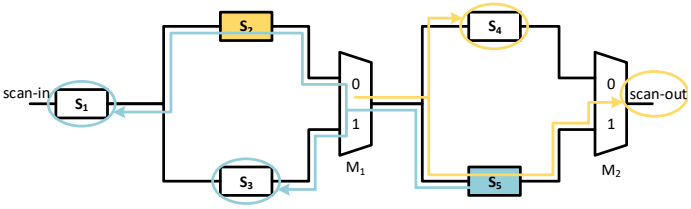

Fig. 4: Potential new predecessors of $\mathtt{S}_5$ and potential new successors of $\mathtt{M}_1$ are chosen by traversing through the RSN.

As shown in Figure 4, the following sets are created for a cut at $\mathtt{M}_1$ and $\mathtt{S}_5$:

$$Succ_{pot}(\mathtt{S}_2) = \{\mathtt{M}_1 \mapsto \mathtt{S}_4,\ \mathtt{M}_1 \mapsto \text{scan-out}\}$$
$$Pred_{pot}(\mathtt{S}_5) = \{\mathtt{S}_1 \mapsto \mathtt{S}_5,\ \mathtt{S}_3 \mapsto \mathtt{S}_5\}$$

Then all change candidates are generated. Every change candidate $\mathcal{C}$ is of the form $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$, where $\mathcal{C}_1$ consists of either one or all elements of $Succ_{pot}(\mathtt{S}_x)$ and similar for $\mathcal{C}_2$. For the example RSN:

$$\mathcal{C}_1 \in \{\{\mathtt{M}_1 \mapsto \mathtt{S}_4\},\ \{\mathtt{M}_1 \mapsto \text{scan-out}\},\ \{\mathtt{M}_1 \mapsto \mathtt{S}_4,\ \mathtt{M}_1 \mapsto \text{scan-out}\}\},$$
$$\mathcal{C}_2 \in \{\{\mathtt{S}_1 \mapsto \mathtt{S}_5\},\ \{\mathtt{S}_3 \mapsto \mathtt{S}_5\},\ \{\mathtt{S}_1 \mapsto \mathtt{S}_5,\ \mathtt{S}_3 \mapsto \mathtt{S}_5\}\}$$

In order to avoid a combinational explosion of possibilities, we do not employ every subset of $Succ_{pot}(\mathtt{S}_x)$ and $Pred_{pot}(\mathtt{S}_y)$ and additionally limit the number of change candidates per security violation to 256. By this method every scan segment is guaranteed to remain in the scan network, and therefore the full scan accessibility is preserved.

Each change candidate $\mathcal{C}$ is then checked by temporarily applying all of its connections, evaluating the changed RSN using a cost function (cf. Chapter IV-C). At last, the change candidate with the lowest cost will be irreversibly applied.

After the application of the change candidate with the lowest cost, the security violation is not necessarily completely resolved yet. In case a security violation is caused by two scan segments that are connected via multiple paths, multiple cuts have to be executed to resolve the security violation. To do so, the method structurally finds another insecure path for the same security violation and the method computes new candidates to resolve this security violation.

C. Evaluating the RSN with a Cost Function

The proposed method supports so-called cost functions to assign any RSN a specific cost, tuning the RSN transformation process toward specific (design) targets, e.g. wiring cost, placement & routing optimization or access latency. We introduce two exemplary cost functions:

1) $cost_{al}$: Determining the shortest structural *access latency* for each scan segment $\mathtt{S}$, where the access latency is measured in traversed scan register bits from the scan-in port over $\mathtt{S}$ to the scan-out port. The cost is then the average over all scan segments. In the original RSN (cf. Figure 1), every scan path traverses $5 + 11 + 11 = 27$ scan register bits, thereby the cost is 27.
2) $cost_{con}$: Each pair of scan segments is called *closely* connected, if they are connected without any other scan register in between. The original RSN has 6 *close* connections: $\mathtt{S}_1 \to \mathtt{S}_2$, $\mathtt{S}_1 \to \mathtt{S}_3$, $\mathtt{S}_2 \to \mathtt{S}_4$, $\mathtt{S}_2 \to \mathtt{S}_5$, $\mathtt{S}_3 \to \mathtt{S}_4$ and $\mathtt{S}_3 \to \mathtt{S}_5$. The cost is then the amount of *close* connection changes compared to the original RSN; naturally, the cost of the original RSN is 0.

In the proposed method, every change candidate is evaluated with the chosen cost function. The change candidate that led to the lowest cost is applied. For the example RSN, the use of $cost_{al}$ leads to the RSN depicted in Figure 5:

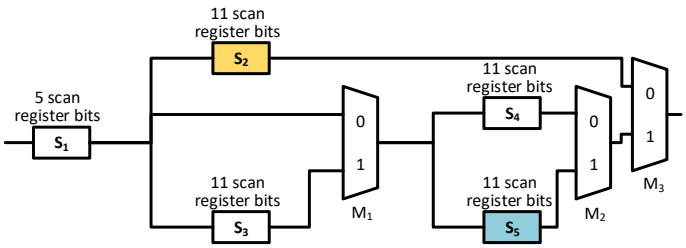

Fig. 5: Secure RSN after execution of the proposed method, utilizing the cost function $cost_{al}$.

In the above RSN the shortest scan path over a scan segment traverses on average only 18.2 scan register bits.

If instead of $cost_{al}$ the cost function $cost_{con}$ is utilized, the change would result in the RSN of Figure 6:

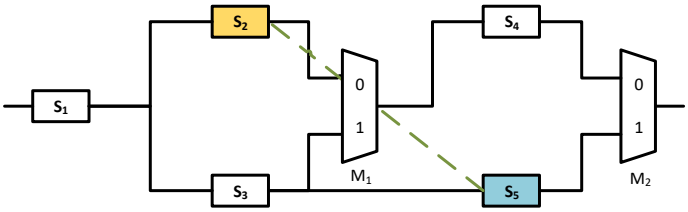

Fig. 6: Secure RSN after execution of the proposed method, utilizing the cost function $cost_{con}$. The removed *close* connection is marked with a dashed line.

The cost of the new RSN is then 1, as the *close* connection between S_2 and S_5 is removed and no new *close* scan segment connections have been created. It can be seen that even for only one design change the choice of the cost function potentially has a big impact on the RSN design.

The proposed method can be flexibly extended by additional structural cost functions such as counting the number of multiplexers or connections between RSN regions. Additionally, we can easily integrate load-profile dependent cost functions such as minimizing the number of cycles needed to apply a given sequence of RSN operations.

After the application of the change, $\overrightarrow{UT}$ and $\overrightarrow{CAT}$ are recomputed, by traversing from every scan segment of a changed connection through the RSN and applying Equations 2 and 3 (cf. Chapter IV-A) until a fixed point is reached.

As soon as the security violation is resolved, the proposed method looks for further un-resolved security violations. If another security violation is found, the above described steps for the corresponding insecure path are executed.

D. Termination, Soundness & Completeness

The method terminates, as the number of possible security violations is finite and each violation is resolved with a finite number of changes, e.g. the cutting of each insecure path for a security violation over multiple paths. The proposed method changes the RSN without causing new security violations, because it only connects segments, that were already connected before via other segments.

For soundness, note that the method terminates with a resulting RSN that does not contain any security violation. Furthermore Lemma 1 (cf. Chapter IV-A) shows that any security violation can be found by locally checking Formulas 4 and 5.

The method is complete, because the simple RSN, where all scan segments are connected in parallel is secure and always reachable, i.e. the method finds a transformation that leads towards that simple RSN. It should be noted, that this simple RSN is more of theoretical nature. In general the proposed method will return a much more feasible RSN.

V. Experimental Evaluation

A. Experimental Setup

The presented methods have been validated and evaluated using the benchmarks introduced by BASTION in [24] which have 10 to 8,485 scan segments. The benchmark set consists of all 17 acyclic benchmarks for which an ICL source file exists.

The algorithm introduced in the previous sections has been implemented on top of the tool *eda1687* [25]. All experiments are conducted on a single core of an Intel Xeon CPU running at 3.3 GHz.

In real world applications the security specification of the scan segments would be individually specified based on the sensitivity and confidentiality of the various employed instruments. In order to allow a thorough evaluation of the characteristics of the presented method on a wide set of benchmarks, we randomly generated the security specification for each benchmark. We distributed three categories of trust and up to seven categories of confidentiality such that a (pessimistic) number of security violations (cf. Table III) occurs.

B. Experimental Results

Table III shows the experimental results for the proposed method utilizing the cost functions $cost_{con}$ and $cost_{al}$ (cf. Chapter IV-C).

In the first column the benchmark names are listed. Columns two to four enlist size information on the corresponding benchmark, namely the total number of scan segments, the total number of scan register bits and the total number of scan multiplexers, respectively. While the total number of scan segments and scan register bits is constant throughout the transformation, the number of scan multiplexers might rise, as scan segments that were in series might be redesigned in parallel. In the fifth column we list the number of scan segments, which cause a security violation with another scan segment in the original RSN; it can be seen that due to the randomized security specification 16.67% (*SoC_DAP_3D*) to 40.61% (*p34392*) of the scan segments cause a security violation with a predecessor.

For each cost function the proposed method was applied on each benchmark; the last 8 columns show information on the runs for $cost_{con}$ and $cost_{al}$, respectively. As described in Chapter IV-B, for each security violation multiple change candidates are evaluated and only the change candidate with the lowest cost is applied. Table III shows both the total number of (evaluated) change candidates and the total number of then applied changes. For the benchmarks *SoC_DAP_3D* and *MultiCoreAL* only one change needs to be applied to retrieve a secure RSN, therefore only a comparably small amount of changes (4 and 48, respectively) needs to be evaluated. Note that even though only two changes were applied to some benchmark RSNs, up to 681 candidates (*Mingle* with $cost_{con}$) were evaluated for these benchmarks. For the largest benchmark *FlexScan*, the method created and investigated over 2,000,000 change candidates out of which around 3,000 were applied for $cost_{con}$ and around 2,200 were applied for $cost_{al}$ to get a secure RSN. Note that the number of applied changes is not identical to the number of security violations as an applied change may resolve multiple security violations. At the same time resolving a security violation may require multiple applied changes.

The Columns *Final Cost* list the cost of the transformed RSN, namely the number of changed *close* connections for the run with $cost_{con}$. For the run with $cost_{al}$ *Final Cost* lists the average shortest access latency, which is on average reduced by 31% compared to the original insecure RSN. Thus, the algorithm was able to reduce the access latency, in addition to securing the RSN. Lastly, the runtime of the proposed method is presented (in seconds). The efficiency of the proposed method is shown, as the complete transformation method runs within a few seconds. It is also seen that the scoring function $cost_{al}$, that requires a global analysis of the RSN structure, takes up more runtime due to its higher complexity, but still remains below 40 seconds, even for the largest benchmark.

Benchmark	#Scan Segments	#Scan Register Bits	#Scan Muxes	#Segm. with security violation	$cost_{con}$				$cost_{al}$			
					#Change Candidates	#Applied Changes	Final Cost	Runtime (in s)	#Change Candidates	#Applied Changes	Final Cost	Runtime (in s)
BasicSCB	21	176	10	5	279	2	5	0.00	209	2	22.90	0.00
Mingle	22	270	13	4	681	2	6	0.00	1 101	4	39.32	0.01
TreeFlat	24	101	24	7	176	2	6	0.00	2 297	7	57.92	0.04
TreeFlatEx	122	5 194	59	37	16 249	33	87	0.02	21 132	50	748.07	0.64
TreeBalanced	90	5 581	46	29	12 647	23	60	0.02	36 615	73	361.61	0.56
TreeUnbalanced	63	41 887	28	25	4 237	11	27	0.01	18 562	56	5,807.70	0.26
q12710	50	26 187	27	19	435	6	17	0.00	475	6	550.14	0.01
t512505	287	77 006	159	96	65 138	94	242	0.08	34 962	85	377.74	3.36
p22810	524	30 118	270	187	154 941	179	454	0.15	50 040	177	162.36	3.44
a586710	64	41 667	32	22	2 798	15	36	0.01	4 607	33	712.30	0.03
p34392	197	23 196	96	80	21 086	65	162	0.03	27 243	130	176.55	0.27
p93791	1 185	98 480	596	403	205 180	349	857	0.19	1 764 448	14 941	183.80	17.13
SoC_DAP_3D	12	216	3	2	4	1	3	0.00	4	1	88.67	0.00
MultiTAP	67	41 894	34	26	21 057	20	53	0.03	4 888	16	763.73	0.12
MultiCoreAccessLink	10	16	8	2	48	1	2	0.00	48	1	10.20	0.00
Kernel	59	22 396	32	17	4 357	9	24	0.01	1 936	6	467.80	0.04
FlexScan	8 485	8 485	4 243	3 012	2 173 724	3 002	7 509	5.19	2 129 705	2 183	2,126.25	36.96

TABLE III: Experimental results invoking the cost function $cost_{con}$ that penalizes changed *close* connections and for the cost function $cost_{al}$ that penalizes *access latency*.

Overall the proposed method returns an RSN, that satisfies the security specification in feasible runtime and fulfills its aim to design the RSN beneficial to specific goals, while maintaining the same scan accessibility as the original RSN.

VI. Conclusions and Future Work

In this work we investigated data flow security violations in reconfigurable scan networks, some of which can only be resolved by design changes, e.g. removing wires and adding wires.

We proposed:

- Structural tests for security violations, where instead of testing each combination of distant scan segments, only local properties have to be checked.
- A sound and complete method to efficiently resolve security violations in an RSN to finally obtain a secure RSN, while maintaining full scan accessibility.
- Cost-based heuristics to effectively tune the method towards scan design targets, e.g. test time or wiring cost.

As demonstrated by the experimental evaluation, the presented method is highly scalable and hence also applicable for transforming large RSN structures toward given scan design goals.

In the future, we plan to extend the algorithm towards integrating design check rules and an optimal synthesis of secure scan networks, e.g. using ILP-based optimization methods.

Acknowledgments

This work was supported by the Baden-Württemberg Stiftung (IKT-Sicherheit, SHIVA).

References

[1] J. Rearick and A. Volz, "A Case Study of Using IEEE P1687 (IJTAG) for High-Speed Serial I/O Characterization and Testing," in *Proc. IEEE Int'l Test Conference (ITC)*, 2006, paper 10.2.

[2] "How to JTAG your Xbox 360 and run homebrew," online: http://www.instructables.com/id/How-to-JTAG-your-Xbox-360-and-run-homebrew (accessed July 14, 2017).

[3] J. D. Rolt, G. D. Natale, M.-L. Flottes, and B. Rouzeyre, "A Novel Differential Scan Attack on Advanced DFT Structures," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 4, pp. 58:1–58:22, Oct. 2013.

[4] M. A. Kochte, M. Sauer, L. Rodriguez Gomez, P. Raiola, B. Becker, and H.-J. Wunderlich, "Specification and Verification of Security in Reconfigurable Scan Networks," in *Proceedings of the 22nd IEEE European Test Symposium (ETS)*, Mai 2017.

[5] D. Hely, M. L. Flottes, F. Bancel, B. Rouzeyre, N. Berard, and M. Renovell, "Scan Design and Secure Chip [Secure IC Testing]," in *Proc. IEEE On-Line Testing Symposium (IOLTS)*, 2004, pp. 219–224.

[6] M. Tehranipoor and C. Wang, Eds., *Introduction to Hardware Security and Trust*. Springer, 2012.

[7] J. Da Rolt, A. Das, G. Di Natale, M.-L. 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, March 2014.

[8] R. Baranowski, M. A. Kochte, and H.-J. 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, 2015.

[9] J. Dworak, A. Crouch, J. Potter, A. Zygmontowicz, and M. Thornton, "Don't Forget to Lock your SIB: Hiding Instruments using P1687," in *Proc. IEEE International Test Conference (ITC)*, 2013, paper 6.2.

[10] 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. Conference on Design, Automation & Test in Europe (DATE)*. EDAA, 2014, pp. 195:1–195:6.

[11] K. Rosenfeld and R. Karri, "Security-Aware SoC Test Access Mechanisms," in *Proc. IEEE VLSI Test Symposium (VTS)*, 2011, pp. 100–104.

[12] G.-M. Chiu and J.-M. Li, "A Secure Test Wrapper Design Against Internal and Boundary Scan Attacks for Embedded Cores," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 20, no. 1, pp. 126–134, Jan. 2012.

[13] G. Sengar, D. Mukhopadhyay, and D. R. Chowdhury, "Secured Flipped Scan-Chain Model for Crypto-Architecture," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2007, pp. 2080 – 2084.

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

[15] R. Nara, H. Atobe, Y. Shi, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "State-Dependent Changeable Scan Architecture against Scan-Based Side Channel Attacks," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 1867–1870.

[16] M. Inoue, T. Yoneda, M. Hasegawa, and H. Fujiwara, "Partial scan approach for secret information protection," in *Journal of Electronic Testing: Theory and Applications*, vol. 27, no. 2, 2011, pp. 99–108.

[17] J. DaRolt, G. D. Natale, M.-L. Flottes, and B. Rouzeyre, "On-chip test comparison for protecting confidential data in secure ICs," in *IEEE European Test Symposium (ETS)*, 2012, pp. 1–1.

[18] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Securing Access to Reconfigurable Scan Networks," in *Proc. IEEE Asian Test Symposium (ATS)*, 2013, pp. 295–300.

[19] ——, "Access Port Protection for Reconfigurable Scan Networks," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 30, no. 6, pp. 711–723, 2014.

[20] A. Atteya, M. A. Kochte, M. Sauer, P. Raiola, B. Becker, and H.-J. Wunderlich, "Online prevention of security violations in reconfigurable scan networks," in *To appear in Proc. IEEE European Test Symposium (ETS)*, 2018.

[21] O. Kömmerling and M. G. Kuhn, "Design Principles for Tamper-Resistant Smartcard Processors," in *Proc. USENIX Workshop on Smartcard Technology (WOST)*. USENIX Association, 1999, pp. 9–20.

[22] E. Ebrard, B. Allard, P. Candelier, and P. Waltz, "Review of Fuse and Antifuse Solutions for Advanced Standard CMOS Technologies," *Microelectronics Journal*, vol. 40, no. 12, pp. 1755–1765, 2009.

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

[24] A. Tšertov, 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 *2016 IEEE International Test Conference (ITC)*, 2016, pp. 1–10.

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