Online Prevention of Security Violations in Reconfigurable Scan Networks

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

Proceedings of the 23rd IEEE European Test Symposium (ETS'18), Bremen, Germany, 28 May - 1 June 2018

doi: https://doi.org/10.1109/ETS.2018.8400685

Abstract: Modern systems-on-chip (SoC) designs are requiring more and more infrastructure for validation, debug, volume test as well as in-field maintenance and repair. Reconfigurable scan networks (RSNs), as allowed by IEEE 1687 (IJTAG) standard, provide flexible access to the infrastructure with low access latency. However, they can also pose a security threat to the system, by leaking information about the system state. In this paper, we present a protection method that monitors access and checks for violations of security properties online. The method prevents unauthorized access to sensitive and secure instruments. In addition, the system integrator can specify more complex security requirements, including giving multiple users different access privileges. Simultaneous accesses to multiple instruments, that would expose sensitive data to an untrusted core (e.g. from 3rd party vendors) or instrument, can be prohibited. The method does not require any change to the RSN architecture and is easily integrable with IP core designs. The area overhead with respect to the size of the RSN is below 6% and scales well with larger networks.

[1] **IEEE COPYRIGHT NOTICE**
©2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Online Prevention of Security Violations in Reconfigurable Scan Networks

Ahmed Atteya[1], Michael A. Kochte[1], Matthias Sauer[2], Pascal Raiola[2],
Bernd Becker[2], Hans-Joachim Wunderlich[1]

[1]ITI, University of Stuttgart, Pfaffenwaldring 47, D-70569 Stuttgart, Germany
[2]University of Freiburg, Georges-Köhler-Allee 51, D-79110 Freiburg, Germany

***Abstract*—Modern systems-on-chip (SoC) designs are requiring more and more infrastructure for validation, debug, volume test as well as in-field maintenance and repair. Reconfigurable scan networks (RSNs), as allowed by IEEE 1687 (IJTAG) standard, provide flexible access to the infrastructure with low access latency. However, they can also pose a security threat to the system, by leaking information about the system state.**

In this paper, we present a protection method that monitors access and checks for violations of security properties online. The method prevents unauthorized access to sensitive and secure instruments. In addition, the system integrator can specify more complex security requirements, including giving multiple users different access privileges. Simultaneous accesses to multiple instruments, that would expose sensitive data to an untrusted core (e.g. from 3rd party vendors) or instrument, can be prohibited. The method does not require any change to the RSN architecture and is easily integrable with IP core designs. The area overhead with respect to the size of the RSN is below 6% and scales well with larger networks.

***Index Terms*—Hardware security, security specification, IJTAG, IEEE Std 1687, reconfigurable scan networks**

I. Introduction

Current systems-on-chip need an ever increasing number of instruments. These instruments could be design-for-test (DfT) structures, sensors, monitors, trace buffers, test and debug controllers or actuators. On-chip instruments are required to facilitate the test, diagnosis and debug required for fast yield bring-up as well as continuous monitoring and maintenance [1, 2]. Thus, the instruments are required to remain operational throughout the lifetime of the chip. The access to this infrastructure through reconfigurable scan networks (RSNs) was standardized in IEEE Std 1687-2014 (IJTAG, [3]).

RSNs allow efficient and flexible access to on-chip infrastructure. By changing the path through which data is shifted, low latency access infrastructure is possible. However, such flexible and interconnected structures also open a side-channel for attacks. Since the infrastructure is integrated into all parts of the system, it is possible for sensitive information to be leaked, to gain unauthorized access and to manipulate the system state, pushing it out of normal safe operation conditions [4–6].

Since attackers can utilize the access port of the RSN, some protection methods attempt to use authentication using cryptographic techniques [7–10], while others present methods based on obfuscation [11, 12]. Memory-mapped instruments such as trace buffers can be secured using [13]. One could also extend the access port with a filter allowing only restricted access to the network [14, 15]. It is possible to extend the RSN with control logic and formally verify that accesses cannot violate security requirements [16, 17].

In [11, 12] locking SIBs (LSIBs), giving access to a lower level of hierarchy, are protected using obfuscation strategies, e.g. distributed key locations and honey traps. A linear feedback shift register (LFSR) and physically unclonable functions (PUFs) were used in [9, 10] to generate the key for an LSIB, this increased the security level compared to obfuscation.

Secure SIBs were presented by [8] that allow access to the protected instruments only after a user has presented a correct response to a randomized challenge. And trustworthy accesses, that shift data only through trusted components, are generated in [18]. It builds upon the concept of trustworthiness presented in [16]. In [14, 15] a sequence filter is presented that restricts the possible RSN accesses (shift in bit sequences) to a static set of precomputed allowed accesses. However, in this way, the access flexibility is reduced.

These methods are not suited for dealing with the increasing complexity of on-chip infrastructure networks and the resulting complex security requirements. Modern chips with multiple possible users, many integrated intellectual property (IP) cores and sensitive instruments need new methods that can fulfill these requirements. The proposed method in this paper is also filter based, but *the model upon which the restriction occurs* is completely different. The proposed method is much more flexible, restricts only the access to sensitive instruments and can enforce more complex security requirements. By representing the relevant structural characteristics and state of the RSN as a finite state machine (FSM) and keeping track of the current configuration, more complex security requirements as in [16], can be specified and enforced, e.g. disallowing accesses based on external control signals or that access sensitive parts of the network. The security requirements are monitored online and concurrent to accesses without any penalty to the latency of allowed accesses. The method can prevent access to a specified subset of instruments, give multiple users different access levels and specify forbidden *simultaneous access* for certain instruments, i.e. accessing multiple instruments at the same time by a specified user can be forbidden to prevent information flow between some cores or instruments.

The next section introduces RSNs and their operation. Section III discusses the introduced security specification that this method enforces and the considered attack model. Section IV discusses the generation of the filter. This is followed by an evaluation of experimental results.

II. Reconfigurable scan networks

An RSN has the following components, signals and operation behaviour. An example of an RSN is shown in Figure 1.

A. Terminology

Scan Segment: consists of a shift register of a certain size n. Data is shifted through scan segments from the scan-in (SI) port to the scan-out (SO) port. An optional shadow register is included for instruments that need bidirectional communication or drive RSN-internal control signals, e.g. mux select signals. The external control signals are *Select, Capture, Shift* and *Update*.

Select Signal: all other control signals will only affect a scan segment if its select signal is activated, i.e. deselected segments do not participate in capture, shift or update operations.

Capture Signal: this signal will trigger the read operation from the instrument to the shift register of the scan segment.

Shift Signal: this signal causes the shift register to start shifting from SI to SO.

Update Signal: if the update signal is asserted the values in the shift register are loaded into the shadow register (if available).

Scan Multiplexer: sets up the path though which data is shifted through the network (usually driven from same source as the select signal of segments on the corresponding path).

Active Scan Path: the configured path of scan segments with activated select signals from the main SI port to the main SO port of the network. Only a single active scan path may exist at a time throughout the network, such that no partial paths or cycles are allowed, i.e. only segments on the active scan path are selected. All other segments are deselected.

Configuration Segment: scan segments that are used to drive control signals to configure the active scan path (either directly or through combinational logic).

Segment Insertion Bits (SIBs): are one bit configuration scan segments that allow access to a lower level, in a hierarchical RSN architecture, when a "1" is stored in the SIB.

B. RSN operation

Reconfigurable Scan Networks are usually accessed through the Test Access Port (TAP) of the JTAG standard [3]. The user keeps track of the active scan path length, the active segments and their positions after every reconfiguration of the RSN.

RSNs are accessed by first capturing data from the active segments, then shifting the length of the active scan path, such that captured data is shifted out while new data is being shifted in, and finally updating the shifted in data. This sequence is called a capture-shift-update (CSU) operation. Any RSN has an initial state that defines a specific active scan path through some scan segments at start-up or when a restart signal is activated. To access other segments that are not in the current active scan path, one or more CSU operations are required. In each CSU operation, the correct configuration values need to be shifted into the configuration segments on the current path to activate the scan segments to be accessed.

An example of an RSN is shown in Fig. 1. The initial configuration of the RSN sets $SI-S1-S3-S4-SIB-SO$ as the active scan path. Assuming all segments store 1-bit, shifting in the pattern "0101" activates both the scan multiplexer $M2$ as well as the SIB, and the new configuration has $SI-S1-S3-S5-S6-SIB-SO$ as the new active scan path.

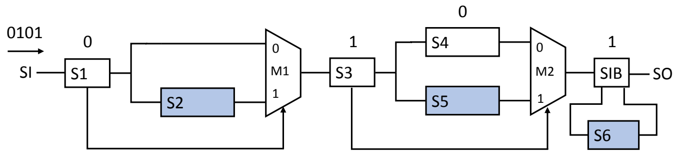

Fig. 1. Example of accessing an RSN with scan multiplexers and SIBs. Segments in blue can be protected.

III. Security of On-chip Infrastructure

In chips with multiple cores from different vendors, where each core has its own integrated infrastructure, security has become a major concern. As shown in [16], security requirements cannot be satisfied by simply blocking access to parts of the infrastructure network anymore. Different users need to have different access rights. For example, dependability procedures for two cores from different vendors should not access any infrastructure outside their corresponding core. Also maintenance technicians should not have the same access rights as testing engineers. Since each core provides different trustworthiness, and the data have different levels of sensitivity, a fine-grained access specification is required.

A. Security Specification

The goal of this method is to prevent any violations to a provided security specification for a given RSN. The specification distinguishes different users or roles that access the RSN, each is given different permissions, e.g. general users, maintenance technicians, test engineers. The RSN contains trusted and untrusted segments, also the data stored in segments can have different levels of required secrecy. Segments that are easily controllable or observable through functional logic must be given an appropriate level of trust, that would prevent *unsafe RSN configurations*, i.e. configurations where sensitive data from a core with high confidentiality is shifted through an untrusted core, or vice versa. The required security specification can be then inferred, using an approach as in [16], and added to the specification to be enforced. The considered security specification is defined as follows.

The set of all scan segments in a given RSN is denoted $\mathcal{R}$ and the set of all possible users to access the RSN is denoted $\mathcal{U}$. Access to all segments is allowed by default, except those specifically restricted by the specification. For the defined properties in the specification, if the restricted users are not specified, then the property is applied to all users $\mathcal{U}$. The set of security specifications is $\mathcal{S} := S_{res} \cup S_{sim}$, where S_{res} and S_{sim} contain properties restricting access to segments and preventing simultaneous access of segments, respectively.

A *restricted segment* is a segment that is not allowed to be a part of the active scan path.

For a user $u_i \in \mathcal{U}$, the set $G(u_i) \subset \mathcal{R}$ defines the segments that u_i is not allowed to access.

Predicate $Active(R_i)$ denotes if a scan segment $R_i \in \mathcal{R}$ is part of the active scan path, i.e. its *select* signal is enabled.

$ActiveUser(u_i)$: Denotes if user $u_i \in \mathcal{U}$ is the current active user.

Restrict access to segments: Is represented by one or more properties $P_i \in S_{res}$. Any segment in $R_i \in G(u_i)$ must not be on the active scan path when user u_i is currently accessing the network.

$$P_i = \forall R_i \in G(u_i) : \neg(Active(R_i) \wedge ActiveUser(u_i))$$

Restrict simultaneous access: To prevent access of multiple groups of segments (e.g. multiple cores) simultaneously for a certain user, the security specification allows to define multiple groups of segments. The system integrator is free to define each group as a single or multiple segments, to forbid the access between single or multiple segments.

$$G_{sim} := \{G_1, ..., G_n | G_i \subset \mathcal{R}, \forall i, j, j \neq k : G_i \cap G_j = \varnothing\}$$

The set of groups $G_{sim}(u_i) \subset G_{sim}$ are restricted from simultaneous access for a user $u_i \in \mathcal{U}$. Each simultaneous access restriction is a property $P_i \in S_{sim}$. Any two segments from different groups $(G_j, G_k) \in G_{sim}(u_i)$ must not be on the active scan path at the same time, while a user u_i is currently accessing the network.

$$P_i = \forall j, k, j \neq k \; \forall R_j \in G_j \; \forall R_k \in G_k : \\ \neg(Active(R_j) \wedge Active(R_k) \wedge ActiveUser(u_i))$$

The specification is enforced by preventing simultaneous accesses, preventing access to a segment or specifying the prohibited user from performing certain accesses. It is enforced using a new type of sequence filters that is based on the relevant characteristics of the RSN structure and state. The filter is placed after the TAP controller as shown in Fig. 2. During an access, new data is shifted in and RSN data is shifted out. If the shifted out data is sensitive or confidential or restricted for the particular user, then the specification already prohibits this configuration ensuring that sensitive data is never exposed.

The authentication of the current user through strong cryptographic codes, e.g. challenge-response protocols or PUFs, is usually enforced by an on-chip security manager [16]. The security manager activates the filter if required and provides it with the needed information about the current user, and the filter informs it when a violation has occurred (Fig. 2). The implementation of the security manager is outside the scope of this paper, however it could be adapted from the implementation presented in [19] or the authentication controller in [8]. It is clear that any secure SoC must implement an authentication scheme for many reasons, as discussed in [13, 19], which can be reused to reduce the overhead.

B. Attacker Model

The following points describe the considered capabilities of an attacker in our approach.

- The attacker has control over the access port (TAP controller).
- The attacker can observe scan data shifted through segments belonging to untrusted cores under his control.

IV. Filter Generation

The proposed filter consists of three main parts (Fig. 2). A finite state machine (*FSM*), a *configuration array* consisting of 1-bit scan segments that hold the current configuration of the RSN and the *synthesized conditions* representing a violation of the security specification. The filter allows only accesses consisting of valid CSU operations, i.e. access operations that capture, then shift the exact length of the currently configured active scan path and finally update. Invalid accesses will be filtered and the update signal will not be asserted. This restriction ensures that the user cannot update unexpected data (e.g. from preceding segments) into the configuration segments of the RSN, causing an unknown or forbidden configuration of the RSN to occur. Patterns for other devices on the same board (but outside the RSN and SoC) must not be shifted through the FSM, to ensure they do not affect the FSM. This could be realized by sending the patterns for the secure SoC directly, e.g. through one of the user-defined instructions and test data registers of the TAP controller.

The FSM tracks which scan segment in the RSN will receive the bit currently being shifted in at SI port, assuming the current access is valid and shifts the complete length of the currently configured active scan path. The FSM also stores and updates the current RSN configuration in the *configuration array*.

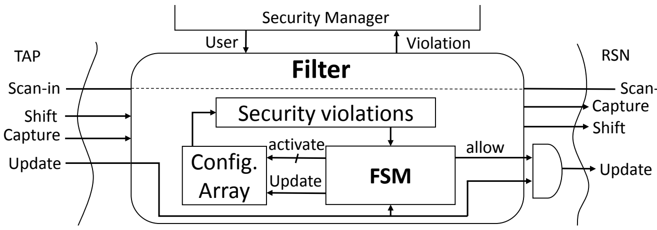

Fig. 2. General architecture of the Filter

The filter has two requirements for ensuring correct functionality.

1) In the initial (reset) configuration of the RSN, all segments on the active scan path must be accessible for any user.
2) Access to all non-restricted segments should be possible, i.e. the union of all paths to restricted segments (for one or all users) must not block access to non-restricted segments.

As an example for these conditions, consider Fig. 1. The segments $S1, S3, S4$ and SIB form the initial configuration of the RSN. Thus, any of these segments cannot be considered as restricted segments. Also according to (2), both $S4$ and $S5$ cannot be protected at the same time, because that would block all further access to SIB as well as leaving no path to the main scan out port. Preventing access to the segments marked in blue is possible as none of them are on the initial configuration and all have other paths to bypass them.

To construct the filter, a structural description of the RSN is parsed and the security specification is provided. A general filter that represents the RSN is generated. Boolean functions that satisfy the provided security specification are generated and synthesized into combinational logic. If an access is detected to violate the security specification, the FSM goes into a *lock* state at the end of the access and prevents an update signal from reaching the RSN. The FSM stays in the *lock* state until a global reset is performed.

A. FSM Generation

The FSM part of the filter gets the following inputs: Scan-in, capture, shift, update, user and violation (indicates security violation, see Section IV-C). The FSM has the following control states:

INIT: The initial state after a reset, and where the FSM waits for a CSU operation to begin. Only the capture control signal may be asserted in this state, attempts to shift or update will cause the FSM to go to the LOCK state.

LOCK: This state can only be left when allowed by the security manager and after a global reset of the entire RSN network. Any violations to the synthesized security requirements or the valid CSU restriction causes the FSM to transition to this state at the end of the access.

UPDATE: The last state of a valid CSU operation. In this state, the *allow* signal (Fig. 2) is activated only if $violation = 0$, permitting the update signal to affect the RSN. The user must update in this state which will then complete the access and transition back to the INIT state, otherwise the FSM considers this a violation of the valid CSU restriction and goes into the LOCK state.

For each 1-bit scan segment in the RSN a state is created. During operation, when the FSM is in a certain state, the bit currently at the main SI port will be stored in the corresponding scan segment after a valid CSU operation is completed. For a valid CSU operation, the first bit to be shifted in will be stored at the last scan segment of the currently configured active scan path, the next will be in the one before it, and so on (Fig. 1). This means that the FSM will transition through the states representing the currently configured active scan path from SO to SI. As such, state transitions occur from the current segment to the structural predecessors in the RSN. Which of the predecessors is chosen depends on the currently configured active scan path, i.e. depends on the values of configuration segments stored in the configuration array. The condition for transition to next state is built from the conjunction of the required control signal (capture, shift or update), the values of the configuration segments controlling the scan multiplexers between the current segment and the target predecessor segment. Fig. 3 shows the FSM for Fig. 1. In the FSM, SIB has three possible predecessors, $S6, S4$ or $S5$. Conditions that a path is generated between each of them are annotated in the figure. It is clear that the condition should contain all possible control values affecting the path between the two segments in the RSN. States for segments connected to the SI port will transition to UPDATE state, which then transitions to LOCK or INIT to block or complete the access.

Finally, after all transitions are generated, a state optimization is performed. Since each state represents a single bit in the actual RSN, there are many redundant states representing the same segment or segments connected serially. If these states do not represent a configuration segment, they are merged using a counter. The values being shifted into the RSN for non-configuration segments will not affect the RSN configuration. For the filter, it is only important to shift the correct length of the segments. The counter is loaded with the length of the segment, then the state is stable until the correct number of shifts has been observed (i.e. counter is 0), and finally the FSM continues transitioning. These final merged states of non-configuration segments represent the smallest granularity (smallest configurable paths in the RSN) that the filter can allow or prevent. Algorithm 1 explains the process of generating the FSM states and the transitions.

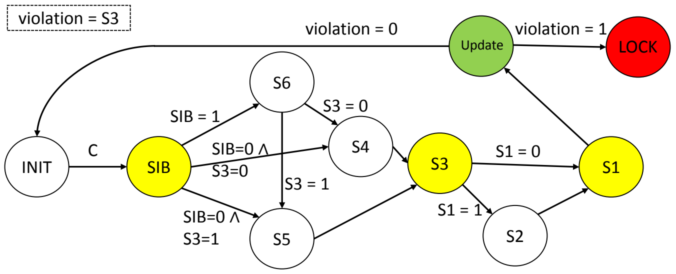

Fig. 3. Generated FSM for RSN in fig. 1

Algorithm 1: FSM Generation Algorithm

```
input : RSN Structural Description
output: FSM
1  create state $Init, Lock, Update$
2  create transition $t_1$: $Update \rightarrow Init$
3  annotate $t_1$ with condition "$violation = 0$" // violation is
     synthesized from security spec.
4  create transition $t_2$: $Update \rightarrow Lock$
5  annotate $t_2$ with condition "$violation = 1$"
6  $stateSet \leftarrow \{Init, Update, Lock\}$
7  foreach 1-bit segment $R_i$ of RSN do
8      $stateSet$.append(new state $S_i$)
9      if $R_i$ is ConfigSegment then
10         $configArray \leftarrow$ new segment $C_i$
11     end
12 end
13 foreach $S_i$ in $stateSet$ do
14     $predecessors \leftarrow \{S_j | R_j$ is predecessor of $R_i\}$
15     foreach $P_i$ in $predecessors$ do
16         create transition $t_i$: $S_i \rightarrow P_i$
17         $cd \leftarrow$ condition that path $(P_i \rightarrow S_i)$ is active
           // The Boolean function of config segments
              that activates the scan path from $P_i$ to
              $S_i$
18
19         annotate $t_i$ with condition $cd$
20     end
21 end
22 merge any two states $S_1, S_2$ that are connected by a single transition
     and $cd = 1$
```

B. Configuration Array

The FSM requires the values of the configuration segments in the RSN. For this purpose, all configuration segments are duplicated into a so called configuration array. During an access, the states transitioned through will always be equivalent to the segments on the actual active scan path of the RSN, assuming the configuration array and the configuration segments in the RSN have the same reset values.

All the segments in the configuration array have both shift and shadow registers. They are all connected to the main SI port as well as the shift and update control inputs. During an access operation, a segment in the configuration array will be selected (activated) if the current state of the FSM represents the corresponding configuration segment in the RSN. Since the definition of current state in the FSM is that the bit at the input SI port will end up at the corresponding segment, this value will also be stored into the shift register of the configuration array. At the end of an allowed valid access, the update signal is propagated to the configuration array, updating the shifted in configuration data to the shadow registers. Since the values in the shadow registers represent the currently configured active scan path, they are used for the transition conditions between states in the FSM. The values in the shift register of the configuration array are used in the synthesized *security violations* because these values represent the access currently being attempted.

C. Synthesis of Security Conditions

The FSM along with the configuration array provide all the information needed about the current access being performed, while allowing only valid CSU operations to be performed. To realize the required security properties, the configuration values of the current access are checked using a Boolean formula that satisfies the security properties defined in Section III-A. The formula(s) are then synthesized into combinational circuits and added to the implementation. The disjunction of all violations to security specification is then fed back to the FSM, instructing it to transition to the LOCK state in case any of the properties was violated.

To restrict access to a specific scan segment (*restricted segment*) for all users, the conditions on the transitions to the state representing this segment need to be all false. Consider the example in Fig. 3. To restrict access to $S5$ the condition $violation = S3$ is synthesized, i.e. $S3$ must never be 1. This *essential condition* can be represented as the intersection of all conditions on the incoming transitions for $S5$. Specifying restrictions for certain users can be trivially extended by adding the disjunction of all restricted users to the generated conditions. Continuing with the previous example, assume only U_1 and U_2 are restricted from accessing $S5$. The condition in this case is $violation = (U_1 \vee U_2) \wedge (S3)$.

Finally, to prevent the simultaneous access to two or more groups of segments, the disjunction of the *essential conditions* of each group is evaluated together, followed by the conjunction of any two groups. For m groups of segments $(G_1, ..., G_m)$ each with n segments $(R_1, ..., R_n)$, a violation to this property is evaluated as follows.

$$ActiveGroup(G_k) = \bigvee_{i=1}^{n} Active(R_i)$$

$$violation = \forall G_a, G_b, a \neq b : \quad ActiveGroup(G_a) \wedge ActiveGroup(G_b)$$

V. Evaluation

To evaluate the effectiveness and cost of the proposed method, two implementations of the ITC'02 benchmarks were used as presented in [18]. The area overhead of the approach was evaluated for restricting access to a subset of the instruments for all users, the restrictions of instruments for several different users, and the prevention of certain simultaneous accesses of instruments.

A. Benchmark Circuits

The benchmarks consist of boundary and internal scan chains connected in a hierarchy either using scan multiplexers as shown in Fig. 4 or using SIBs. The benchmarks consist of several modules, and each module has a group of segments. The number of MUXes/SIBs, number of scan registers, number of scan bits as well as the area of the synthesized benchmarks are presented in the first section of Table I.

The MUX-based benchmarks have two types of access. Configuration access where configuration segments are accesses to add or remove certain segments from the active scan path, and data accesses where the segments configured onto the scan path are then accessed. The SIB-based benchmarks, on the other hand, have distributed control for accessing lower levels of the hierarchy.

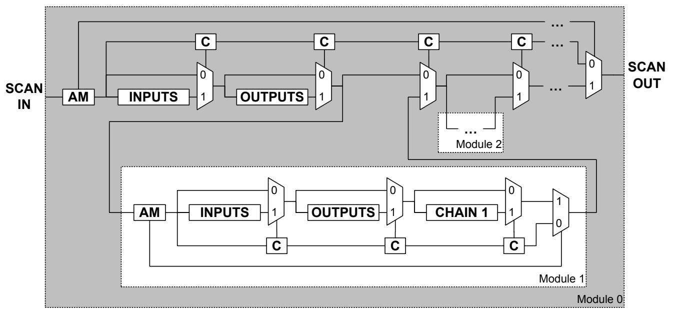

Fig. 4. Architecture of MUX implementation of ITC benchmark [18]

B. Experiments

Three main experiments were performed in this evaluation. In *experiment (1)*, simple access restriction to specific instruments for all users was performed. Different runs of the experiment were performed, randomly restricting access to 25%, 50% and 75% of the instruments in each benchmark. Configuration scan segments are not targeted for protection, only regular instruments are protected. In *experiment (2)*, four different users are defined with different access privelages. One user is allowed to access all segments while the others were restricted from accessing 25%, 50% and 75% of the available segments in the benchmark. Finally in *experiment(3)*, two random modules in the benchmark are chosen. A subset of segments from one module was restricted from being accessed at the same time as any register in the second module.

Using a commercial unbounded model checker, the protection method was formally verified at RT-level. Due to the difficulty of formally verifying RSNs, as discussed in [17], the verification could only be performed on a small RSN implementation.

All the benchmarks as well as all generated filters were synthesized using the Nangate 45nm open cell library with area optimization target. The filter can be clocked up to 300 MHz, and has no negative impact on circuit timing since the shift frequency is typically only a fraction of this value. The resulting overheads for all experiments are shown in Table I. The table shows the area of the benchmarks, their characteristics and the resulting number of states (configuration and total) for the resulting filter. The number of states in the filter increases with the number of configurable paths in the RSN. The area overhead of the filters also depends on the complexity of the conditions for transitioning between states. These conditions can be much more complicated in MUX based benchmarks (e.g. Fig. 4) as the possible paths (and their corresponding activation conditions) grow proportionally to the number of MUXes and paths that need to be configured to connect the two segments in the active path. The overhead for SIB benchmarks is much lower due to the simpler structure and the lower number of possible connections for each segment.

The last three sections of Table I show the area of the synthesized filters and their overheads for the three experiments.

TABLE I
AREA AND CHARACTERISTICS OF SYNTHESIZED BENCHMARKS AND FILTERS

Design	Benchmark characteristics				Filter	75% seg. restriction		Different users		Simul. access restriction	
	Area $[\mu m^2]$	#MUX /#SIBs	#scan seg	#scan bits	# states config/total	Area $[\mu m^2]$	Overhead [+%]	Area $[\mu m^2]$	Overhead [+%]	Area $[\mu m^2]$	Overhead [+%]
f2126 mux	221 754	45	81	15 834	45/84	2 179	0.98%	2 179	0.98%	2 175	0.98%
f2126 sib	221 614	41	77	15 830	40/79	1 297	0.59%	1 305	0.59%	1 295	0.58%
q12710 mux	367 949	30	51	26 188	30/54	1 280	0.35%	1 287	0.35%	1 278	0.35%
q12710 sib	367 813	25	47	26 183	25/49	913	0.25%	920	0.25%	913	0.25%
p22810 mux	425 946	311	565	30 139	311/568	25 492	5.98%	25 500	5.99%	25 426	5.97%
p22810 sib	425 035	283	537	30 111	282/539	10 768	2.53%	10 771	2.53%	10 735	2.53%
p34392 mux	327 407	142	245	23 261	142/248	8 491	2.59%	8 494	2.59%	8 449	2.58%
p34392 sib	326 854	123	226	23 242	122/228	3 622	1.11%	3 624	1.11%	3 593	1.10%
p93791 mux	1 382 172	653	1241	98 637	653/1244	71 754	5.19%	71 779	5.19%	71 628	5.18%
p93791 sib	1 381 081	621	1209	98 605	620/1211	22 322	1.62%	22 345	1.62%	22 203	1.61%
t512505 mux	1 080 353	191	319	77 037	191/322	14 833	1.37%	14 826	1.37%	14 815	1.37%
t512505 sib	1 079 456	160	288	77 006	159/290	5 286	0.49%	5 289	0.49%	5 270	0.49%
a586710 mux	589 510	47	79	41 682	47/82	2 179	0.37%	2 181	0.37%	2 165	0.37%
a586710 sib	589 304	40	72	41 675	39/74	1 298	0.22%	1 302	0.22%	1 287	0.22%

In *experiment (1)*, the results for restricting 25% and 50% of the segments are very close to that of restricting 75% (less than 0.01% difference) and were omitted due to space restriction. For all experiments, the area overhead is clearly very low compared to the area of the RSN itself, with the maximum overhead being 5.99% for the p22810 mux benchmark. Most benchmarks have an overhead of less than 2%. It is also clear that the method scales well with an increasing number of specified security requirements. A maximum of 0.02% overhead is incurred when increasing the number of monitored security properties in our experiments. This holds for increasing the number of restricted segments, adding different user privileges and also blocking accesses to different cores at the same time. When all security requirements are combined, only the additional synthesized conditions grow, which incurs a small additional overhead (around 0.05%). This shows the capability of our approach to handle complex systems and scale well with increasing number of security specifications.

VI. CONCLUSION

While infrastructure access has become a requirement throughout the lifetime of any modern SoC, it also poses a security threat, possibly leaking sensitive data. As the size and complexity of this infrastructure increases, security requirements with higher complexity are needed. In this work, a new type of filters for reconfigurable scan networks (RSNs) is introduced as a protection method. The filter is placed at the interface of the TAP controller. It requires no architectural changes to the RSN, which facilitates its usage with verified IP core designs. The filter is transparent to accesses that do not violate any specified security properties. Modeling the RSN structure in the filter allows more complex security properties to be monitored during operation, e.g. different user privileges, or prohibiting simultaneous activation of instruments. The area overhead of the filter ranges from 0.1% to 6% when compared to the original RSN area, and scales well with larger circuits.

ACKNOWLEDGMENTS

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

BIBLIOGRAPHY

[1] N. Stollon, *On-Chip Instrumentation: Design and Debug for Systems on Chip*. Springer US, 2011.

[2] F. G. Zadegan, E. Larsson, A. Jutman, S. Devadze, and R. Krenz-Baath, “Design, Verification, and Application of IEEE 1687,” in *Proc. IEEE Asian Test Symposium (ATS)*, Nov 2014, pp. 93–100.

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

[4] J. Dworak and A. Crouch, “A call to action: Securing IEEE 1687 and the need for an IEEE test Security Standard,” in *Proc. IEEE 33rd VLSI Test Symposium (VTS)*, April 2015, pp. 1–4.

[5] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. Springer Publishing Company, Incorporated, 2011.

[6] D. Mukhopadhyay, S. Banerjee, D. RoyChowdhury, and B. B. Bhattacharya, “CryptoScan: A Secured Scan Chain Architecture,” in *Proc. IEEE Asian Test Symposium (ATS)*, Dec 2005, pp. 348–353.

[7] K. Park, S. G. Yoo, T. Kim, and J. Kim, “JTAG Security System Based on Credentials,” *Journal of Electronic Testing*, vol. 26, no. 5, pp. 549–557, Oct 2010.

[8] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, “Fine-Grained Access Management in Reconfigurable Scan Networks,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 34, no. 6, pp. 937–946, 2015.

[9] S. K. K, N. Satheesh, A. Mahapatra, S. Sahoo, and K. K. Mahapatra, “Securing IEEE 1687 Standard On-chip Instrumentation Access Using PUF,” in *2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, Dec 2016, pp. 56–61.

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

[11] 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)*, Sept 2013, pp. 1–10.

[12] A. Zygmontowicz, J. Dworak, A. Crouch, and J. Potter, “Making it harder to unlock an LSIB: Honeytraps and misdirection in a P1687 network,” in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–6.

[13] J. Backer, D. Hely, and R. Karri, “Secure and flexible trace-based debugging of systems-on-chip,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 2, pp. 31:1–31:25, Dec. 2016.

[14] 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.

[15] 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.

[16] M. A. Kochte, M. Sauer, L. R. Gomez, P. Raiola, B. Becker, and H. J. Wunderlich, “Specification and Verification of Security in Reconfigurable Scan Networks,” in *22nd IEEE European Test Symposium (ETS)*, May 2017, pp. 1–6.

[17] M. A. Kochte, R. Baranowski, M. Sauer, B. Becker, and H.-J. Wunderlich, “Formal Verification of Secure Reconfigurable Scan Network Infrastructure,” in *Proc. 21st IEEE European Test Symposium (ETS)*, 2016, pp. 1–6.

[18] M. A. Kochte, R. Baranowski, and H. J. Wunderlich, “Trustworthy reconfigurable access to on-chip infrastructure,” in *International Test Conference in Asia (ITC-Asia)*, Sept 2017, pp. 119–124.

[19] A. P. D. Nath, S. Ray, A. Basak, and S. Bhunia, “System-on-chip security architecture and cad framework for hardware patch,” in *23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2018, pp. 733–738.