Specification and Verification of Security in Reconfigurable Scan Networks

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

Proceedings of the 22nd IEEE European Test Symposium (ETS'17) Limassol, Cyprus, 22-26 May 2017

doi: http://dx.doi.org/10.1109/ETS.2017.7968247

Abstract: A large amount of on-chip infrastructure, such as design-for-test, debug, monitoring, or calibration, is required for the efficient manufacturing, debug, and operation of complex hardware systems. The access to such infrastructure poses severe system safety and security threats since it may constitute a side-channel exposing internal state, sensitive data, or IP to attackers. Reconfigurable scan networks (RSNs) have been proposed as a scalable and flexible scan-based access mechanism to on-chip infrastructure. The increasing number and variety of integrated infrastructure as well as diverse access constraints over the system lifetime demand for systematic methods for the specification and formal verification of access protection and security properties in RSNs. This work presents a novel method to specify and verify fine-grained access permissions and restrictions to instruments attached to an RSN. The permissions and restrictions are transformed into predicates that are added to a formal model of a given RSN to prove which access properties hold or do not hold.

Specification and Verification of Security in Reconfigurable Scan Networks

Michael A. Kochte[1], Matthias Sauer[2], Laura Rodriguez Gomez[1], 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*—A large amount of on-chip infrastructure, such as design-for-test, debug, monitoring, or calibration, is required for the efficient manufacturing, debug, and operation of complex hardware systems. The access to such infrastructure poses severe system safety and security threats since it may constitute a side-channel exposing internal state, sensitive data, or IP to attackers.**

Reconfigurable scan networks (RSNs) have been proposed as a scalable and flexible scan-based access mechanism to on-chip infrastructure. The increasing number and variety of integrated infrastructure as well as diverse access constraints over the system lifetime demand for systematic methods for the specification and formal verification of access protection and security properties in RSNs.

This work presents a novel method to specify and verify fine-grained access permissions and restrictions to instruments attached to an RSN. The permissions and restrictions are transformed into predicates that are added to a formal model of a given RSN to prove which access properties hold or do not hold.

***Keywords*-Access Control, On-Chip Infrastructure, Reconfigurable Scan Network, Verification, Side-Channel Attack, IEEE Std 1687, IJTAG, Hardware Security**

I. Introduction

Current systems-on-chip and 3D-ICs integrate an increasing amount of infrastructure with different on-chip instruments to facilitate test and test control, diagnosis, post-silicon validation, debug, bring-up, or maintenance. Access to this infrastructure is required during manufacturing, bring-up and post-silicon validation of the system, in the field, even concurrent to operation for monitoring or reliability management, and at the end-of-life, for instance for diagnosis of field returns [1, 2, 3].

This growing amount and diversity of on-chip instrumentation requires flexible and scalable access mechanisms. Reconfigurable scan networks (RSNs) meet these requirements and have been recently standardized in IEEE Std. 1149.1-2013 and IEEE Std. 1687-2014.

The access to on-chip infrastructure constitutes a side-channel for attacks, resulting in leakage of confidential information, IP theft, or system manipulation, as demonstrated by popular hardware attacks exploiting the unprotected JTAG access [4, 5]. Disabling the access to the infrastructure, for instance by disconnection of the RSN interface after manufacturing using a wafer saw or e-fuses [6, 7], is not attractive since accessibility is still required throughout the system lifetime.

Depending on the operation mode and accessing user, different access permissions to the components of the infrastructure are required. During diagnosis or post-silicon validation, unlimited access to all components is necessary. During operation, it should not be allowed to activate for instance the built-in self-test (BIST) of a component in a safety-critical system. On the other hand, when the system undergoes maintenance in a workshop and is set in a test mode, the BIST controller must be accessible again. Infrastructure accesses concurrent to the operation, e.g. by power-management firmware, shall be restricted to those structures that do not interfere with the operation and safety and security requirements.

The contradiction between observability and controllability required in complex systems and safety and security requirements have been already addressed for conventional design-for-test infrastructures such as static scan chains, e.g. in [8, 9, 10]. Most of the solutions focus on the application to crypto cores and limit the possibility to extract information about the encryption key or process by authorization or obfuscation of structure or data.

In [11, 12], secure test wrappers are proposed that en-/decrypt test data on a per-core level to protect test access from untrusted cores or attackers. Verification of formal security properties were used e.g. in [13, 14, 15] to indicate the absence of hardware Trojans [9].

An overview of security threats and countermeasures for JTAG-based infrastructure access is given in [16]. In [17], a secure JTAG controller is described with four different access levels that are set statically by blowing e-fuses. In the JTAG architectures in [18, 19], protected scan chains can be accessed after authentication.

The structure of RSNs differs significantly from JTAG-based access, requiring novel architectures for access protection. In [20, 21], access protection is implemented based on obfuscation, and in [22] based on authorization using strong cryptography-based authentication for fine-grained access control. The filters at the RSN interface in [23, 24] limit the allowed access sequences to a set of precomputed ones.

To verify the correctness of secure RSN architectures and find design bugs or flaws in the architecture, verification based on formal methods is required. For instance, verification can be used to prove that protected scan registers are inaccessible to unauthorized users. For general RSNs, bounded [25] and unbounded [26] model checking has been proposed. To overcome the complexity due to high sequential depth, efficient domain-specific models with abstracted temporal resolution are used.

Complex systems with diverse infrastructure require a design methodology for security and access management. This comprises the consideration of the different aspects of security such as accessibility of components and resources for authorized entities, integrity (absence of unauthorized system alterations), and confidentiality (no unauthorized disclosure of information) [27]. In addition, the trustworthiness of system components, i.e. the confidence that a component performs as expected [27], must be taken into account when the system security is assessed. Regarding the on-chip infrastructure in such systems, these security attributes must be implemented by methods of access control, information isolation, encryption, and assurance of integrity of data and internal state.

This work proposes for the first time a method to specify access permissions and prohibitions to components in the on-chip infrastructure. We describe how specified permissions and restrictions are transformed into predicates for a formal model of the RSN that is used for infrastructure access. Based on the model, we demonstrate how to prove whether the specified access properties hold in a given RSN.

The following section briefly summarizes the structure and modeling of RSNs. Section III describes the targeted security properties and requirements. The threat model and the verification method is explained in Section IV, followed by a case study in Section V.

II. Preliminaries

A. Reconfigurable Scan Networks (RSNs)

The building blocks of RSNs are scan segments, scan multiplexers, and control logic. The principal element is a scan segment, consisting of a shift and an optional shadow register that interfaces to instruments or other mission logic. A read or write access to segments in the RSN consists of the capture phase (reading data from attached instruments), the shift phase, and the update phase (writing shifted data into instruments). Such an access is also called a *CSU operation*.

A scan segment can be multiple bits wide and has signals to control that the segment (1) is enabled for capture, shift, and update operations (*select* signal); (2) does not participate in the update operation regardless of the select signal (*updis* signal); (3) does not participate in the capture operation regardless of the select signal (*capdis* signal). During the shift phase, data is shifted through the shift register of the segment. During the update phase, data is stored in its shadow register.

Scan multiplexers allow to change the segments through which data is shifted. A scan path is a path from the primary scan input through scan segments and scan multiplexers to a primary scan output. A scan path is called *active* if all segments on the path are *select*ed and participate in the shift operation.

The state of control and multiplexer address signals is determined by external control inputs, the state of shadow registers of arbitrary scan segments, and combinational functions thereof. The state of the scan segments is called *scan configuration*. A scan configuration is called valid if it defines an active scan path and the scan segments that are not on that path are de*select*ed. RSNs are typically accessed via a JTAG test access port (TAP). Additional details can be found in IEEE Std. 1687 or in [25].

B. CSU-Accurate Modeling of RSNs

For efficient formal modeling of an RSN, we employ the *CSU-accurate RSN model* (CAM) introduced in [25] and summarized below. The CAM is constructed as an abstract finite state machine, in which a transition corresponds to a complete CSU operation with multiple cycles of operation in the actual RSN.

The CSU-accurate model $\mathbb{M} = \{S, I, C, c_0, T\}$ consists of the set of state elements S, the set of control signals I, the set of scan configurations $C \subseteq \{0,1\}^{|S \cup I|}$, the initial scan configuration $c_0 \in C$, and the transition relation $T \subseteq C \times C$. A state element $s \in S$ models a 1-bit shadow register of a scan segment in the RSN. A scan configuration $c \in C$ specifies the state of all elements in S and control inputs in I. It is also interpreted as a function $c : S \cup I \mapsto \{0,1\}$ mapping each element $e \in S \cup I$ to state $c(e)$.

The transition relation T includes the pairs of scan configurations $c_1, c_2 \in C$ such that c_2 is reachable from c_1 by one CSU operation. The characteristic function of T is defined as:

$$T(c_1, c_2) := \bigwedge_{s \in S} \left[(\texttt{Active}(c_1, s) = 0) \rightarrow (c_2(s) = c_1(s)) \right]$$

The predicate $\texttt{Active} : C \times S \mapsto \{0,1\}$ assigns each element $s \in S$ and configuration $c \in C$ a Boolean value which is true ($\texttt{Active}(c, s) = 1$) if and only if s is selected in c and c is a *valid* scan configuration, i.e., when s belongs to the active scan path in c.

In [25], the concept of *robust* RSNs is discussed, in which every reachable configuration is a *valid* one. For robust RSNs, the set of reachable states in the CSU-accurate model is equivalent to the reachable states in the cycle-accurate model. This allows efficient, sound and complete formal verification. Robustness of an RSN can be efficiently checked by SAT-based induction [25].

C. Unbounded Model Checking using Craig Interpolation

Model checking (MC) can be used to compute if a given target property, such as the accessibility of a certain scan segment, is reachable in a sequential circuit independent of the number of time steps. The behavior of the circuit and the property are encoded as a propositional formula of the form:

$$MC_k := I_0 \wedge T_{0,1} \wedge \ldots \wedge T_{k-1,k} \wedge P_k, \qquad (1)$$

where I_0 encodes the set of initial states and $T_{i,i+1}$ represents the transition function modeling the temporal step of the system from time frame i to $i+1$. Finally, P_k represents the target property, which needs to hold after k time steps.

Such a formula is solved by iterating over the number of considered time frames and checking if the property P_k does hold after k iterations. If P_k does hold, a trace (sequence of state transitions) is generated that leads from an initial state to a state in which the property holds. Proving the unreachability of the property requires to generate a fixed point of reachable states, i.e., to show that each reachable system state has been considered without reaching the target.

Craig interpolation is an efficient method to compute this fixed point by overapproximation of reachable states [28]. In this work, we employ the in-house Craig interpolation based *CIP* solver [29] since it has been tuned to work well on digital logic and provides a trace (counter-example) if P is reachable.

III. Security Properties and Requirements

This section describes the proposed security properties and requirements in the RSN of the system infrastructure and their modeling. The specification of security in complex systems calls for fine granularity and comprises three main aspects to be considered:

Authorization: Accesses to on-chip infrastructure are only possible if the accessing entity is authorized.

Confidentiality: Sensitive, protected data is not exposed to potential leaky channels. Leaky channels comprise untrusted components in the system or side-channels that may allow data observation via external interfaces.

Integrity: Sensitive, protected data cannot be modified by attackers in the system. This entails that data shifted in an RSN cannot be spoofed and RSN access operations cannot be disturbed by untrusted components.

Figure 1 shows the considered system architecture. We assume the existence of a security manager providing information on the currently active user (after authentication) and current system mode to the RSN. Based on these values, the control logic in the RSN allows or prohibits accesses to certain scan segments.

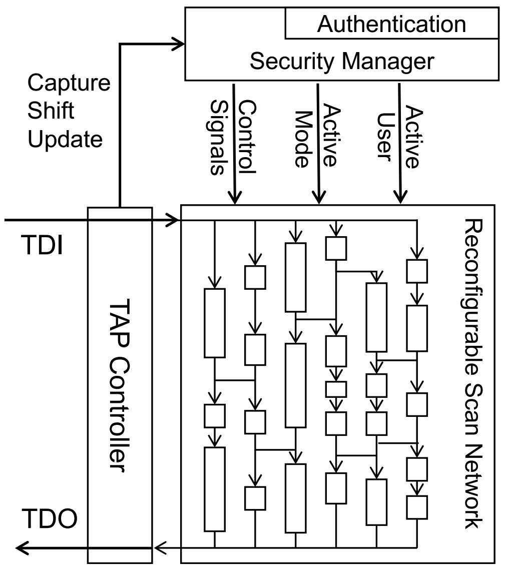

Figure 1. Overview of the architecture consisting of the test access port (TAP), the security manager, and the reconfigurable scan network.

The security and access management in the RSN are described in terms of:

Components in the RSN: Scan segments or registers R_i. The set of all scan segments is denoted $\mathcal{R}$.

Operation mode M_i describes a mode of the system operation. $M_i \in \mathcal{M} \subseteq \{\text{system, test, debug}\}$, for instance, denotes that the system can be in system (online) mode, test mode, or debug mode.

User or role U_i in the RSN, that accesses infrastructure components $\mathcal{R}$. A user may be allowed or prohibited to read or write to components depending on the current mode M_i.

The following *security properties* of the system are specified by the system integrator at component level:

Trustworthiness: $Trust(R_i)$ of scan segment R_i is a measure of how trustworthy R_i (or its surrounding core) in the RSN is. A component with low trust threatens security by sniffing (leaking) or spoofing transmitted information. Components have low trust if for instance the source of the IP of the core / component is not trusted (insecure third party designs), or if it can be easily observed or controlled via interfaces (ease of accessibility).

Data confidentiality: $C(R_i)$ is a measure of data sensitivity, i.e., the level of secrecy or required protection of the data stored in or read from a component R_i in the RSN. *Confidential data shall not be exposed to untrusted components.* We assume that the levels of data confidentiality are ordered.

Data integrity requires that shifted data is not modified by unauthorized entities or components on the scan path. This is typically achieved by message authentication codes, and in JTAG architectures by de/encryption at the TAP [16]. Checking the authenticity of data locally at each accessed segment in the RSN entails high area cost. Here, we address data integrity within the RSN by shifting data only through trusted components.

In addition, the system-level integrator specifies the following *security requirements*, which define:

- the RSN accesses that user U_i is authorized or prohibited to perform. Access A_i is a tuple $(R_i, O_i, M_i) \in \mathcal{R} \times \{read, write, read\&write\} \times \mathcal{M}$ that describes the read or write access to a scan segment R_i in operation mode M_i. When necessary, an authorized access is denoted A_i^+, a prohibited access is denoted as A_i^-. In case an access is not explicitly authorized or prohibited, we assume the access is *don't care* and hence may or may not be possible.
- a mapping between each level of data confidentiality C and a set of levels of trustworthiness: $CT : C \mapsto \wp(Trust)$. This mapping describes the required trustworthiness of components through which data of confidentiality C may be transmitted (shifted).

The following rules govern how an access A_i to the target segment R_i may be performed by user U_i and how data may be transmitted (shifted) in the RSN so that the security requirements are not violated. To read or write data from segments in the RSN, an active scan path from the primary scan-in port to the primary scan-out port over other scan segments is defined. The configuration of such a path may require additional RSN accesses, so that an access results in a sequence of CSU operations, each of which configuring one active scan path $p_1, ..., p_n$. Let

$$P_i := \{R' \mid R' \in \mathcal{R} \text{ is on the active scan path } p_i\}$$

denote the set of scan segments on the active scan path p_i. Data is then shifted through all scan segments P_i.

For each active scan path, the confidentiality and integrity of shifted information must be ensured. For this purpose, the maximum degree of confidentiality required by the data in the scan segments on scan path p_i is computed as: $C_{max}(P_i) := \max_{R' \in P_i}\{C(R')\}$. For each active scan path, all segments on that path must have the required level of trust corresponding to C_{max}:

$$\forall R' \in P_i : Trust(R') \in CT(C_{max}(P_i)).$$

User U_i must be authorized to access all segments on the scan paths configured during access A_i.

IV. Verification of Security Properties

The objectives of our verification method are to

- prove that there is no CSU access sequence that violates the security specification (confidentiality),
- prove that for each allowed access, there exists a CSU access sequence to the target segments (accessibility),
- prove that none of the prohibited accesses is possible.

To prove or refute the security specification in a given RSN, its parts are transformed into constraints for a formal RSN model. For robust RSNs, the verification is sound and complete when the abstract CSU-accurate model from Section II-B is used. This section explains the threat model, the mapping to the RSN model and the verification flow for robust RSNs.

A. Threat Model

In this work, we consider attacks at the logical level, i.e., the attacker has access to the RSN interface (TAP) and to untrusted components. Via the TAP, the attacker can apply arbitrary accesses to the RSN and observe the shift-out data. The observation or control of data and control signals other than those of untrusted components is considered to be beyond the capabilities of the attacker. We assume that the authorization mechanism and security manager are not compromised by e.g. hardware Trojans or a flawed implementation.

B. Mapping of Security Properties and Requirements to the formal RSN Model

For the mapping of the security specification to the formal RSN model, we construct constraints that consider the trust requirements of transmitted data in the RSN, depending on the level of confidentiality of accessed data, the user and mode and specified accesses. To this end, we employ the following variables or sets of variables:

$Active(R_i)$: Each scan segment R_i is assigned a Boolean variable $Active(R_i)$ that denotes whether the segment's *select* control signal is asserted and hence the segment participates in a CSU operation.

$Capture(R_i)$: Each scan segment R_i is assigned a Boolean variable $Capture(R_i)$ that denotes whether the segment participates in the capture operation, i.e., its *capdis* control signal (cf. Sec. II-A) is deasserted.

$Update(R_i)$: Each scan segment R_i is assigned a Boolean variable $Update(R_i)$ that denotes whether the segment participates in the update operation, i.e., its *updis* control signal is deasserted.

$Outcone(R_i)$: This set contains each scan segment in the transitive output cone of R_i, i.e., the scan segments to which a structural scan path from R_i exists.

$Incone(R_i)$: This set contains each scan segment in the transitive input cone of R_i, i.e., the scan segments from which a structural scan path to R_i exists.

$ReqTrust(R_i)$: This set contains the required trust levels of R_i as defined by $C(R_i)$ and the mapping CT from $C()$ to $Trust$.

$ActiveUser(U_i)$: A Boolean variable that is asserted if and only if U_i is the current user.

$ActiveMode(Mode_i)$: A Boolean variable that is asserted if and only if $Mode_i$ is the current system mode.

$Target(A_i)$: Denotes the target scan segment $R_i \in \mathcal{R}$ of an access A_i

$Operation(A_i)$: Denotes the access type ($read$, $write$ or $read\&write$) of an access A_i

$Mode(A_i)$: Denotes the mode $M_i \in \mathcal{M}$ of access A_i.

Using these variables, we express the security properties and requirements as follows:

Confidentiality: For all $R_i \in \mathcal{R}$, the following invariant holds:

$$\forall R_j \in Outcone(R_i) \cup Incone(R_i) : (Active(R_i) \wedge Active(R_j)) \rightarrow Trust(R_j) \in ReqTrust(R_i),$$

i.e., the segments R_j on the active scan path through R_i must be sufficiently trustworthy for the transmitted data.

Existence of allowed accesses: For each allowed access A_i^+ of user U_i, there must exist a CSU sequence in which the following constraints hold at least once:

$$(Operation(A_i^+) \in \{read,\ read\&write\}) \rightarrow (ActiveUser(U_i) \wedge ActiveMode(Mode(A_i^+)) \wedge Capture(Target(A_i^+)))$$

$$(Operation(A_i^+) \in \{write,\ read\&write\}) \rightarrow (ActiveUser(U_i) \wedge ActiveMode(Mode(A_i^+)) \wedge Update(Target(A_i^+)))$$

No prohibited accesses: For each prohibited access A_i^- of user U_i, the following constraints must hold for every CSU sequence:

$$(ActiveUser(U_i) \wedge ActiveMode(Mode(A_i^-)) \wedge Capture(Target(A_i^-))) \rightarrow (Operation(A_i^-) = write)$$

$$(ActiveUser(U_i) \land ActiveMode(Mode(A_i^-)) \land$$
$$Update(Target(A_i^-))) \rightarrow (Operation(A_i^-) = read)$$

This means that if a CSU access to the target of A_i^- exists for the particular user and mode, then the operation may not be the prohibited one.

C. Verification Process

Based on the previously described modeling, we verify that (1) the allowed accesses are possible, (2) the prohibited accesses are not possible, and (3) that no access violates data confidentiality.

In order to do so, we construct a model checking (MC) instance for the CIP solver requiring that $Capture(Target(A_i))$ (or Update) has been asserted once in the sequence and $Active(Target(A_i))$ is enabled in the final state. In case (1), we expect a resulting sequence that proves the access possibility, while in case (2) such a sequence should not exist. In the MC instance, the initial states are defined by the reset values of the scan segments. The transition relation is extracted from the RSN. For robust RSNs, which we consider here, the security verification can be performed very efficiently using CSU-accurate modeling (cf. Sec. II-B).

An allowed access may not be possible under the security specification if for instance all scan paths through a target segment include a segment which the user is not authorized to access or which requires higher trustworthiness than the target. In such cases, the contradiction between the RSN and the security specification can be resolved by relaxing the security requirements or changing the structure of the RSN.

D. Extensions to the Model

The presented verification flow is highly flexible and can be extended in different ways.

Depending on the granularity of the specified operation modes and users, it may be beneficial to allow multiple of them to be present at a given time. One application could be a scenario where the RSN supports a fine-grained group-based authentication system where each group is granted the access rights of multiple users. Our verification flow is able to support such a scenario by defining a set of variables representing the currently active role and by enforcing the variables representing the currently active user according to its membership of the active role.

Additionally, we assumed that an access is don't care if not explicitly allowed or prohibited to remove the burden of unnecessary definitions for the system designer. One can easily implement more complex rules that generate the allowed or prohibited sets of accesses by for example denying all accesses to components with a confidentiality level greater than a particular level for a given user. Also, opt-in or opt-out rulesets that define access rules for not explicitly specified accesses are easily possible. In order to do so, such high level rules need to be converted to its fundamental accesses and can hence be easily implemented in our flow.

V. Case Study

The following case study illustrates the specification of security properties and the described verification method on a small example. The verification is implemented in our C++ based framework for RSN analysis. The considered RSN, shown in Figure 2, is motivated from an automotive context. It has three scan segments RA, RB, RC to reconfigure the scan path, and eight segments $R1$ to $R8$ that interface to instrumentation. As shown in the figure, the function of these segments includes sensors, parameters, or debug and BIST control.

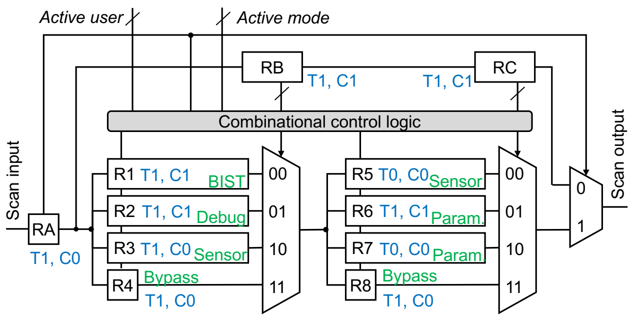

Figure 2. Case study: RSN with scan segments with different trustworthiness and data confidentiality.

In this example, we consider two levels of trustworthiness $T0, T1$ and two levels of data confidentiality $C0, C1$. Confidential data, labeled $C1$, requires a trust level of $T1$. Segments $R5$ and $R7$ are classified as not trustworthy because the source of their IP is third party or their value can be observed from an external interface. This implies, for instance, that an active scan path cannot include $R1$ and $R5$ at the same time, since confidential data from $R1$ may not be exposed to untrusted segment $R5$.

For this RSN, we consider three users *OEM, WS,* and *CPU* representing the OEM, a workshop engineer, and online access by the CPU. We distinguish the system mode M_S and a test mode M_T. For the sake of brevity, we assume that both read and write access is granted (A^+) or prohibited (A^-) in the respective operation modes for the segments as listed below:

CPU (Online)	A^+: $M_S\{R3, R4, R5, R8\}$
	A^-: $M_S\{R1, R2, R6, R7\}$
	A^-: $M_T\{R1, R2, R3, R5, R6, R7\}$
WS (Workshop)	A^+: $M_T\{R1, R3, R4, R5, R6, R7, R8\}$
	A^-: $M_T\{R2\}$
	A^-: $M_S\{R1, R2, R3, R5, R6, R7\}$
OEM	A^+: $M_T\{*\}$, M_S: $\{*\}$

Registers RA, RB, RC are accessible for all users in all modes. The bypasses $R4, R8$ are not explicitly prohibited for user *CPU* in test mode and user *WS* in system mode.

For this specification, we synthesized the control logic in the RSN, which comprises the multiplexer address signals and the control signals of the segments, such that the resulting RSN is *robust* (cf. Sec. II-B). The one-hot encoded signals *Active user* and *Active mode* indicate the currently active user and mode. These signals are controlled by the authentication or security manager of the system (cf. Fig. 1). The control logic in the resulting RSN ensures that the security requirements and access specification are not violated. If a user tries to access segments for which access has not been granted or which violates data confidentiality, a bypass will be selected in order to reach a valid scan configuration with an active scan path. This ensures robustness of the RSN and completeness of the verification when using a CSU-accurate model. The automated synthesis of such control logic for complex infrastructure is beyond the scope of this paper.

For each of the specified allowed and prohibited accesses as well as for the verification of the data confidentiality, a verification step is performed. Such a step consists of the construction of the MC instance and the invocation of the CIP solver. In this case study, 66 verification steps are required. The maximum number of clauses in the MC instances is 540. The overall runtime is less than one second including the parsing of the RSN (specified in IEEE Std. 1687 ICL format).

VI. Conclusion

Systems with complex on-chip infrastructure require fine-grained access management to the infrastructure to satisfy safety and security requirements. This work introduces a method to specify and formally verify security properties and requirements for systems with reconfigurable scan networks as infrastructure access mechanism. The security specification addresses data confidentiality, integrity, and access authorization depending on the accessing user and operation mode. A case study demonstrates the application of the method to an example from the automotive domain.

Acknowledgements

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

[3] E. Larsson and K. Sibin, "Fault management in an IEEE P1687 (IJTAG) environment," in *Proc. IEEE Int'l Symp. Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2012, embedded tutorial.

[4] "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 Nov. 15, 2016).

[5] L. Greenemeier, "iPhone Hacks Annoy AT&T but Are Unlikely to Bruise Apple," *Scientific American*, Aug. 30, 2007, online: http://www.scientificamerican.com/article/iphone-hacks-annoy-at (accessed Nov. 15, 2016).

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

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

[8] D. Hely, M. L. Flottes *et al.*, "Scan Design and Secure Chip [Secure IC Testing]," in *Proc. IEEE Int'l On-Line Testing Symposium (IOLTS)*, 2004, pp. 219–224.

[9] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. Springer, 2011.

[10] J. Da Rolt, A. Das *et al.*, "Test versus Security: Past and Present," *IEEE Trans. on Emerging Topics in Computing*, vol. 2, no. 1, pp. 50–62, Mar. 2014.

[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 (VLSI) Systems*, vol. 20, no. 1, pp. 126–134, Jan. 2012.

[13] X. Zhang and M. Tehranipoor, "Case Study: Detecting Hardware Trojans in Third-Party Digital IP Cores," in *Proc. IEEE Int'l Symp. Hardware-Oriented Security and Trust (HOST)*, 2011, pp. 67–70.

[14] J. Rajendran, V. Vedula, and R. Karri, "Detecting Malicious Modifications of Data in Third-Party Intellectual Property Cores," in *Proc. Design Automation Conference (DAC)*, 2015, pp. 112:1–112:6.

[15] X. Guo, R. G. Dutta *et al.*, "Scalable SoC Trust Verification using Integrated Theorem Proving and Model Checking," in *IEEE International Symposium on Hardware Oriented Security and Trust, HOST*, 2016, pp. 124–129.

[16] K. Rosenfeld and R. Karri, "Attacks and Defenses for JTAG," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 36–47, 2010.

[17] A. Ashkenazi, "Security Features in the i.MX31 and i.MX31L Multimedia Applications Processors," Jun. 2005, freescale Whitepaper IMX31SECURITYWP.

[18] K.-Y. Park, S.-G. Yoo *et al.*, "JTAG Security System Based on Credentials," *Journal of Electronic Testing (JETTA)*, vol. 26, pp. 549–557, 2010.

[19] L. Pierce and S. Tragoudas, "Enhanced Secure Architecture for Joint Action Test Group Systems," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 7, pp. 1342–1345, 2013.

[20] J. Dworak, A. Crouch *et al.*, "Don't Forget to Lock your SIB: Hiding Instruments using P1687," in *Proc. IEEE International Test Conference (ITC)*, 2013, paper 6.2.

[21] A. Zygmontowicz, J. Dworak *et al.*, "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.

[22] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Fine-grained access management in reconfigurable scan networks," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 937–946, June 2015.

[23] 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.

[24] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Access Port Protection for Reconfigurable Scan Networks," *Journal of Electronic Testing (JETTA)*, vol. 30, pp. 711–723, 2014.

[25] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Reconfigurable Scan Networks: Modeling, Verification, and Optimal Pattern Generation," *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 2, pp. 30:1–30:27, 2015.

[26] M. A. Kochte, R. Baranowski *et al.*, "Formal Verification of Secure Reconfigurable Scan Network Infrastructure," in *Proc. IEEE European Test Symposium (ETS)*, 2016, pp. 1–6.

[27] A. Avizienis, J.-C. Laprie *et al.*, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

[28] K. McMillan, "Applications of craig interpolants in model checking," in *TACAS*, ser. LNCS, N. Halbwachs and L. Zuck, Eds. Springer, 2005, vol. 3440, pp. 1–12.

[29] S. Kupferschmid, M. Lewis *et al.*, "Incremental Preprocessing Methods for Use in BMC," *Formal Methods in System Design*, pp. 1–20, 2011.