2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

An Automated Pre-silicon IP Trustworthiness Assessment for Hardware Assurance

John Hallman, David Landoll
OneSpin Solutions, San Jose, CA, USA ({first.last}@onespin.com)

Sergio Marchese, Sven Beyer
OneSpin Solutions, Munich, Germany ({first.last}@onespin.com)

Garrett Chan, Salam Zantout, Vikram Rao
The Aerospace Corporation, El Segundo, CA, USA ({first.last}@aero.org)

Abstract—**Integrated circuit designs include in-house and third-party intellectual properties that could contain hardware Trojans. An independent, trusted, and complete IP model, suitable for automated formal comparison with the IP register-transfer level (RTL) code using commercially available tools, may be used to prove absence of functional Trojans. Such models are generally not available, except for certain critical IPs, as, for example, RISC-V processor cores. The development of these models may be costly and time-consuming. This paper proposes an IP trustworthiness assessment process that does not require a trusted model. The approach uses automated tools that scan the IP RTL code to detect suspicious or unusual code patterns and known Trojan signatures. This low-effort, objective assessment may detect Trojans and provide warnings that, depending on the specific project circumstances, may require additional investigation. The approach is demonstrated on numerous open-source and proprietary test designs containing hardware Trojans.**

Keywords— trusted integrated circuits; IC; hardware Trojans; hardware assurance; security; semiconductor IP.

I. INTRODUCTION

All phases in the development, fabrication, and distribution of integrated circuits (ICs) are vulnerable to the actions of malicious actors. In-house and third-party semiconductor intellectual properties (IPs) used during IC development are no exception. They may contain hardware Trojans inserted in the register-transfer or gate-level models [1][8][10]. Pre-silicon functional verification at the system-on-chip (SoC) or IP level has little chance of detecting Trojans [4][11]. Traditional and advanced verification approaches mainly target intended use cases. These flows are not fit for the purpose of identifying deliberately hidden functionality. Post-silicon Trojan detection methods [9] are valuable. However, a robust and efficient hardware assurance process must be tightly integrated with the development flow, enabling detection of issues as soon as possible. Late detection can result in much higher costs and sets off alarm bells on the quality of previous assurance steps.

A. *Related Work*

Pre-silicon detection of hardware Trojans has attracted interest in academic circles for several years [4][5]. The semiconductor industry, on the other hand, is only recently becoming concerned about this type of attack. There is a lack of dedicated processes, expertise, and commercial, industrial-scale technology. Methods have been proposed that could allow engineers to prove absence of functional Trojans [3]. However, these methods rely on the availability of a suitable, independent model of the IP. Such a model is often not available and too costly to develop. The investment required may not be justified considering the risk level associated with the IP and the assurance level required by the end application.

B. *Contribution*

This paper introduces a pre-silicon trustworthiness assessment process for hardware assurance. Crucially, this process does not require an additional model of the IP. The approach uses electronic design automation (EDA) technology that is commercially available and that scales to large, complex designs. The approach can be integrated into existing ASIC and FPGA development flows and used by engineers with no specific Trojan detection expertise.

The rest of the paper is organized as follows. Section II discusses the application scope of the approach. Section III explains the key aspects of the approach and presents its capabilities and limitations. Section IV presents the results of the application of the approach to several open-source and proprietary designs containing hardware Trojans. Section V concludes the paper.

II. TOWARD TROJAN-FREE ICS

Code review is a common, valuable approach to assess the quality of an IP and complement other hardware assurance techniques. However, engineers integrating an IP into a SoC typically have little knowledge of its implementation details. Following the intricacies of highly-optimized, complex IPs with many configuration options is a daunting task. SoC integrators need a low-effort, automated process to gain confidence that the IPs they use do not contain hardware Trojans. Independent assurance engineers carrying out this process cannot be expected to have intimate knowledge of the IP or have extensive Trojan detection expertise. The results of the assessment process must be easy to understand and communicate to management. Ultimately, the assessment process might unequivocally detect the presence of a Trojan, or report potential issues, group them into categories, and enable a severity grading. Depending on the results and the specific circumstances of the project, it might be necessary to perform additional analysis. Some crucial considerations are:

- Where has the IP been sourced from? An IP developed in house may be deemed less concerning than one obtained from “open source”, licensed by a third-party, or developed by a subcontractor.
- What is the role of the IP in the system? An IP with critical control functions, for example, may require a higher level of trustworthiness.
- Is the IP reused across multiple SoCs? A processor core integrated into multiple SoCs, for example, could be worth additional analysis.

Considering the IP context, if the outcome of the assessment is not satisfactory, engineers must have the means to perform additional analysis and debug any issue reported. The goal of this follow-on analysis is to determine whether an issue flagged by the assessment process is associated with a Trojan, thus requiring further action, or if it can be deemed innocuous and waived. Because of the potential challenges in the follow-on analysis, the availability of pre-packaged trust assurance solutions for specific IPs should also be considered. A solution for trust assurance of RISC-V cores, for example, is presented in [3]. Unlike a generic trust assessment process, which can only provide confidence in Trojan absence, an IP-specific solution could achieve proof of Trojan absence.

The next section presents an IP trustworthiness assessment solution that satisfies many of the requirements discussed in this section.

III. AUTOMATED IP TRUSTWORTHINESS ASSESSMENT

The trustworthiness assessment process presented in this section uses proprietary automated design analysis technology to examine the IP’s RTL code (see Figure 1). The process uses the RTL model in order to establish confidence early in the design cycle. This early design phase offers one of the easiest entry points for an adversary to infiltrate a design with malicious code. Addressing issues at the RTL prevents their propagating to other phases of the design cycle. The confidence achieved at the RTL design phase can be maintained in subsequent design implementation steps. Technologies such as formal logic equivalence checking provide check points and verify that the design maintains its integrity through transformations such as synthesis and place and route.

Given the IP’s RTL model and a script with tool commands to compile the design and set up the process, a report is generated listing design information and a set of issues that could be caused by the presence of Trojans. Reliability issues, such as dead or redundant code, which could be exploited for malicious purposes in later design phases, are also reported. Additional information and capabilities enable a detailed analysis of each reported issue. Issues can be reviewed and prioritized according to an estimated severity level. Further investigation, should that be deemed necessary, may focus on high-severity issues.

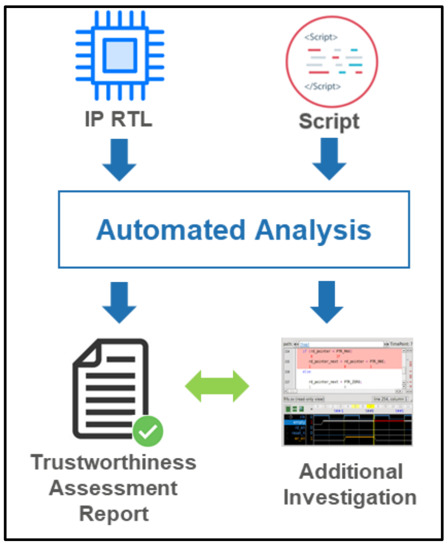


Figure 1. IP trust assessment using automated design analysis technology that generates a report listing potential issues and debug information for additional analysis.

A. Trigger and payload detection

Design analysis focuses on detecting suspicious or unusual code structures or functions that could implement a Trojan trigger or payload [14]. Examples include a counter activating a function when reaching a defined value, a specific sequence of values in the data path influencing a control function, a finite state machine (FSM) that could enter a deadlock state, and redundant logic.

B. Technology

For effective automated trust assessment, it is necessary to use a variety of both new and established design analysis techniques. Advanced structural analysis can be used to partition the design and identify portions of logic that could be influenced by specific signals. Structural analysis is leveraged for both detecting suspicious logic and reducing the complexity of other analysis steps. Formal engines may be deployed to exhaustively exercise certain sequential behaviors of the design and identify or rule out corner cases. For example, automatic assertion generation and formal proofs used in advanced functional verification environments to detect deadlock conditions and other types of issues can also be applied for trust assurance. Finally, domain-specific data, based on known Trojan signatures, can also be leveraged in detection algorithms.

C. Issues and limitations

The initial approach uses a less aggressive strategy in reporting suspicious trigger structures in order to reduce the number of false alarms. Noise greatly reduces the effectiveness of the approach, as each flagged item in the design necessarily requires at least some additional analysis to confirm or refute the presence of nefarious logic. Conversely, a more aggressive strategy can be used when confidently identifying reliability issues. These potential vulnerabilities, when not addressed, offer the opportunity for an attacker to exploit a design without being detected. In the initial experiments, the number of these potential vulnerabilities may be excessive and create the need for filters or other automated process handling. A balance among these strategies will be essential for extracting the most utility out of the reporting capability.

A limitation of this approach is that it does not provide proof of Trojan absence, which is a very demanding goal. When no issues are detected using this approach, the design may still contain Trojans. However, the process can ensure that designs are free from certain classes of Trojans characterized by the type of triggers and payloads that are scanned for, thus increasing the assurance level.

While the runtime is short for most of the designs analyzed, in some instances, design complexity issues have led to long runtimes or even inconclusive results. It is often possible to resolve complexity issues, but that requires additional engineering effort and expertise.

IV. Results

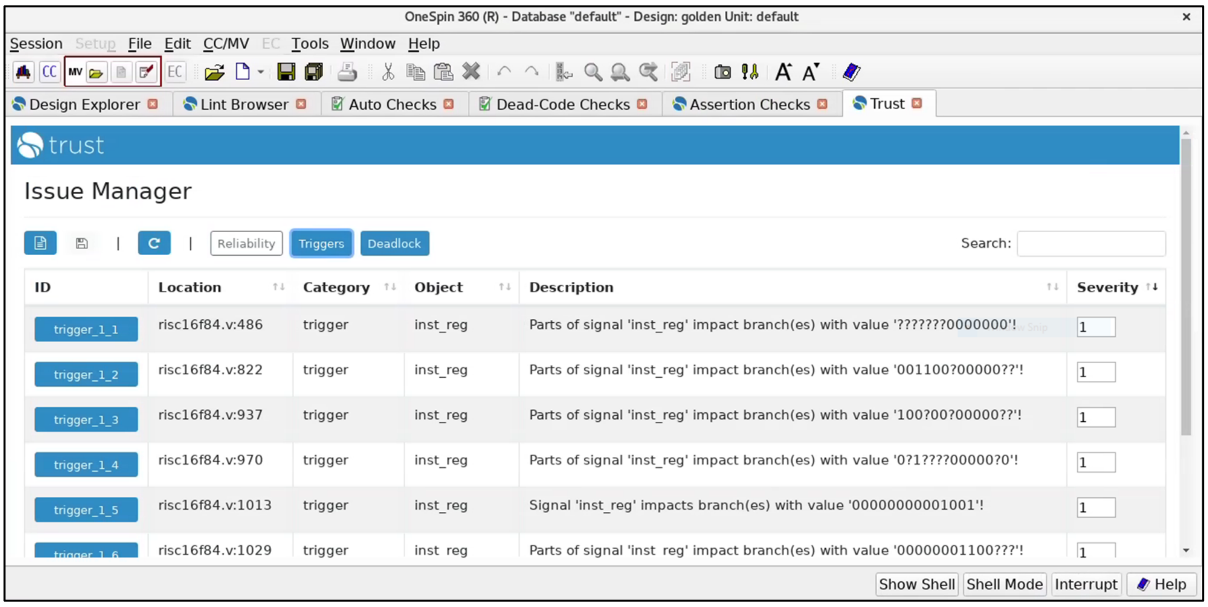
The IP trustworthiness assessment process outlined in this paper has been applied to 90 RTL designs taken from various sources, including TrustHub [2], GitHub [12], OneSpin generated, and Aerospace provided. The designs cover a large variety of coding patterns and IPs, including processor cores and subsystems (e.g., a RISC-V Rocketcore with memory, SPI, and AXI4 interface IPs). Many designs have been modified to insert hardware Trojans. The approximate number of design state elements ranges from 100 to 100K flip-flops.

The process has been executed by OneSpin Solutions [6] and The Aerospace Corporation [13] engineers, using EDA technology from OneSpin Solutions.

As part of the Aerospace Corporation's hardware assurance efforts, Aerospace performed an evaluation of OneSpin's tool to determine its effectiveness in hardware Trojan detection. Three host designs, SpaceWire, RISC-V Taiga, and LEON3, were chosen to represent real-world MIL/Aero IPs of varying size and complexity. Six hardware Trojan triggers and four payloads were developed (coded) based on Aerospace's hardware Trojan catalog and taxonomy, as well as academic literature. Nine Trojans were created from the combinations of triggers and payloads described in Table 1 and inserted into the host designs. The three "golden" host designs (without Trojans) served as the baseline. The versions of the host designs with the Trojans inserted were used to evaluate the tool against the baseline. OneSpin provided a TCL setup script template, which required minimal modification to run the tool. The script follows OneSpin's conventional flow, in which the user specifies the HDL design files and reset sequence, and then executes the elaborate and compile commands. The tool was executed on each of the designs, and a trustworthiness assessment report (see Figure 2) was generated. The report includes all the issues (potential Trojans) detected. For each issue, the tool also reports (with a link) the related location in the design and a short description, which supports further investigation to determine if the issue is a legitimate hardware Trojan.

Table 1. Types and descriptions of triggers and payloads used in Aerospace's test articles

Trigger Type	**Description**
Always On	No hardware Trojan trigger
Data Input Sequence	Trojan is activated when a specified input sequence is observed by the design
Time Based Counter	Trojan is activated when a specified time is reached
Event Counter	Trojan is activated when an event occurs a specified number of times
Pattern Match	Trojan is activated when the specified input is observed by the design
Partitioned	Trigger is divided into sub-triggers that are ANDed with the output of the malicious function. Trojan is activated when all sub-triggers are satisfied at the same time
Payload Type	**Description**
Data Corruption	Data in design is modified to become unusable
Denial of Service (DoS)	Design functionality is disrupted temporarily or permanently
Privilege Escalation/De-escalation	Higher-level of permissions obtained by unauthorized user
Protocol Violation	Design behavior deviates from specified protocol described in the associated standards document


OneSpin 360 (R) - Database "default" - Design: golden Unit: default

Issue Manager

ID	Location	Category	Object	Description	Severity
trigger_1_1	risc16f84.v:486	trigger	inst_reg	Parts of signal 'inst_reg' impact branch(es) with value '???????0000000'!	1
trigger_1_2	risc16f84.v:822	trigger	inst_reg	Parts of signal 'inst_reg' impact branch(es) with value '001100?00000??'!	1
trigger_1_3	risc16f84.v:937	trigger	inst_reg	Parts of signal 'inst_reg' impact branch(es) with value '100?00?00000??'!	1
trigger_1_4	risc16f84.v:970	trigger	inst_reg	Parts of signal 'inst_reg' impact branch(es) with value '0?1????00000?0'!	1
trigger_1_5	risc16f84.v:1013	trigger	inst_reg	Signal 'inst_reg' impacts branch(es) with value '00000000001001'!	1
trigger_1_6	risc16f84.v:1029	trigger	inst_reg	Parts of signal 'inst_reg' impact branch(es) with value '0000000110???'!	1

Figure 2. Screenshot of the trustworthiness assessment report generated by the OneSpin tool. For each issue, the report includes a unique identifier (first column) that can be clicked to visualize the associated IP source code, an attribute that classify issues in different categories, a description, and a default severity level. The user can modify the severity level and add comments and waivers while reviewing issues.

The tool runtime was recorded from the initial reading of the HDL design files to the generation of the report. The runtime increased in conjunction with the design size and complexity. The RISC-V Taiga design, consisting of 63K flops (with 58K of those flops related to RAM), had a 30-minute runtime, whereas the LEON3, composed of 80K flops used to implement several IP cores (e.g., Ethernet, SpaceWire, AHB Controller), had an approximate runtime of 3 hours. Aerospace collected the following data for each run:

- Design type
- Design complexity
- Number of states (flip-flops)
- Number of FSMs
- Number of counters
- Trojan type (trigger/payload)
- Runtime
- Number of issues (total/trigger/reliability)
- Trojan detected (yes/no)
- False positives

The data collected by Aerospace was provided to OneSpin as part of an independent evaluation. A summary of the collected information is included in the collection of experimental results reported in Table 2. The evaluation results are being be used to inform further discussion between The Aerospace Corporation and OneSpin in order to enhance the Trojan detection capabilities of the tool.

Table 2. Summary of automated trustworthiness assessment results

Source	Name	Runtime	Issues Reported	Trojan Inserted	Automatic Detection
TrustHub*	AES	11 hr	260	Yes	Yes
TrustHub	PIC16	< 1 min	72	Yes	Yes
TrustHub	RS232	< 1 min	3	Yes	Yes
TrustHub	BasicRSA	< 1 min	17	Yes	Yes
GitHub	RISC-Rocketcore	28 min	12	No	No
OneSpin	UART	< 1 min	10	Yes	Yes
Aerospace**	SpaceWire	< 1 min	3	No/Yes	No
Aerospace	RISC-V Taiga	13 min	46	No/Yes	No
Aerospace	LEON3	6 hr	423	No/Yes	Yes***

*Averaged results of multiple test articles

**Includes one golden design, and three derivative designs with Trojans

***Includes three designs with Trojans, one of which is detected

A. Report

The trustworthiness assessment report presented to the IP auditing engineer is concise and easy to understand and review. The report provides a list of reliability issues and other issues that could be due to the presence of a Trojan. Each issue has a unique identifier (first column), a location in the RTL source code, a category, a description, and a default severity level that can be modified by the user, among other fields. Issues can be searched, filtered, and reordered.

For designs in this project, the tool has reported reliability issues (up to 2500). Manual review and waiving large numbers of issues is a time-consuming task. However, reliability issues may be grouped into categories, speeding up the review process significantly. Reliability issues include warnings about dead or redundant RTL code that could be exploited in subsequent design implementation steps, such as synthesis, for Trojan insertion. It is worth noting that in safety-critical projects, all RTL code should be associated with functional requirements or otherwise justified or removed. Therefore, in order to perform a rigorous trustworthiness assessment, reliability issues must not be ignored.

The tool also identified potential Trojan triggers (in a range of 0 to 15 per design). Crucially, the number of issues identified is low, and the number of false alarms is very low (less than 100 for the entire test suite). Even in the worst-case scenario, the additional effort required to analyze the reported issues is manageable.

Table 2 reports results and information for a selection of 9 design groups. The designs have been chosen arbitrarily, but with the aim of representing the entire test suite and range of results obtained. Each entry in Table 2 includes the design's source, name, whether it contains an intentionally inserted Trojan, the tool runtime, the number of issues identified, and whether at least one identified issue detects the Trojan (when one is present).

B. Debug

When selecting an issue's identifier, a new window displays a source code tracing tool highlighting the RTL statements causing it. It is also possible to automatically generate input stimuli and visualize signal waveforms for more in-depth analysis. For a trigger-type issue, the tool can generate multiple sets of input stimuli that cause the potential Trojan trigger to be activated. It is worth noting that achieving the same objective using the traditional approach of a simulation testbench would require significant effort and in-depth knowledge of the IP.

C. Detected Trojans

As shown in Table 2, the tool detects Trojans in many different designs by identifying suspicious code with hidden trigger characteristics. In many cases, the triggers rely on deep counters or state machines with malicious logic that monitor specific input data sequences. For more information on the type of Trojans considered, please refer to [2]. A detailed analysis of the type of Trojans inserted by the authors is beyond the scope of the paper.

D. Missed Trojans

The tool failed to detect the Trojans present in 10 designs. A common thread among these designs is their use of Trojan triggers based on counters that count basic control events. The challenge in these specific cases is that the activation code follows a common coding pattern. Without any knowledge of the expected functional behavior of the design, it is difficult for algorithms to identify and detect this code as suspicious without generating many false alarms. However, when considering Trojans, it is also important to estimate how stealthy they are. A Trojan that is not very stealthy, for example, always triggering on a sequence of events, might be hard to detect automatically with our approach, but would likely be triggered on a basic system verification test, increasing the chances of it being discovered. Nevertheless, the missed detections demonstrate that Trojan detection algorithms need continuous improvement.

E. False alarms

The tool has reported a low number of false alarms, with respect to trigger detection. In a set of related designs, a class of code functions were marked as suspicious Trojan triggers, when in fact they were implementing legitimate functions. One of these cases is related to a test access port (TAP) controller using a specific sequence of data and address inputs to take control actions. In general, this is indeed an unusual, suspicious behavior that would meet our criteria for reporting as an issue.

V. CONCLUSION

The process presented in this paper allows ASIC, FPGA, and SoC developers and integrators to increase confidence that an IP design is trustworthy and free from hardware Trojans. The approach is highly automated and does not require additional models or in-depth knowledge of the IP.

Results demonstrate that the design analysis step requires a one-time process of minutes to hours of computational runtime for industrial-scale IP designs. The process efficiently identifies different classes of trigger circuitry and reliability issues (or vulnerabilities/weaknesses), which could be leveraged to introduce Trojans in subsequent design implementation steps. The reported issues will focus on those functions in the RTL source code that require attention, while minimizing excess noise.

Engineers can perform additional analysis of the issues identified to confirm the presence of a hardware Trojan and its location. The issues reported are linked to the RTL source code that has caused them in order to facilitate code inspection. The tool also automatically generates a signal waveform trace showing how a particular input stimulus triggers a potential Trojan. The visualization of a counterexample has proven to be quite valuable when debugging and fixing the identified issue.

The trustworthiness assessment process represents a low-effort approach to gain confidence of Trojan's absence in IPs. However, it is not yet sufficient to rigorously prove Trojan absence. For an alternative process aiming at proving Trojan's absence, please refer to [3]. Moreover, the process does not yet provide an objective measure of trustworthiness. Trustworthiness metrics are the subject of future work that will leverage industry-wide initiatives. It is important to note that while the complete automation was unable detect all Trojans, this experiment demonstrated a process where the automation contributes valuable data and analysis. Similar to antivirus software programs, automated Trojan detection technology requires continuous improvements to minimize the risk of missing a Trojan, while also keeping noise levels and runtime acceptable.

REFERENCES

[1] C. Krieg, C. Wolf, A. Jantsch, and T. Zseby, “Toggle MUX: How X-Optimism Can Lead to Malicious Hardware,” Design Automation Conference (DAC), 2017.

[2] TrustHub, https://www.trust-hub.org/benchmarks/chip-level-trojan [accessed July 9th, 2020].

[3] D. Landoll, S. Marchese, S. Beyer, N. Tusinschi, P. McHale, V. Bhattacherjee, “Complete Formal Verification of RISC-V Processor IPs for Trojan-Free Trusted ICs,” GOMACTech Conference, 2019.

[4] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmark development," IEEE Int. Conference on Computer Design (ICCD), 2013.

[5] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, “Benchmarking of Hardware Trojans and Maliciously Affected Circuits,” Journal of Hardware and Systems Security (HaSS), April 2017.

[6] OneSpin Solutions, https://www.onespin.com [accessed April 5th, 2020].

[7] User Manual: OneSpin 360™ Version 2018.2.2, OneSpin Solutions.

[8] J. Zhang, F. Yuan, and Q. Xu, “DeTrust: Defeating Hardware Trust Verification with Stealthy Implicitly-Triggered Hardware Trojans,” Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Pages 153-166.

[9] J. Dubeuf, D. Héli, and R. Karri, “Run-Time Detection of Hardware Trojans: The Processor Protection Unit,” 18th IEEE European Test Symposium (ETS), 2013.

[10] X. Wang, T. Mal-Sarkar, A. Krishna, S. Narasimhan, and S. Bhunia, “Software Exploitable Hardware Trojans in Embedded Processors,” IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012

[11] I. Tripathi, A. Saxena, A. Verma, and P. Aggarwal, “The Process and Proof for Formal Sign-off - A Live Case Study,” Design and Verification Conference (DVCon), 2016

[12] GitHub, https://github.com [accessed April 6th, 2020].

[13] The Aerospace Corporation, https://aerospace.org [accessed July 9th, 2020].

[14] TrustHub, https://www.trust-hub.org/downloads/resource/pdf/Taxonomy.pdf [accessed July 9th, 2020].