DFTT: Design for Trojan Test

Yier Jin*, Nathan Kupp*, and Yiorgos Makris†
*Department of Electrical Engineering, Yale University
†Departments of Electrical Engineering and Computer Science, Yale University

***Abstract*— Due to the globalization of the Integrated Circuit (IC) manufacturing industry, hardware Trojans constitute an increasingly probable threat to both commercial and military applications. As traditional testing methods fall short in finding hardware Trojans, several specialized detection methods have surfaced. To facilitate research in this area and embed internal barriers to prevent Trojan attacks both at the design level and at the manufacturing level, we propose a Design-for-Trojan-Test (DFTT) methodology. DFTT is based on one key principle: increase the complexity for hardware Trojan attackers, thereby making successful hardware Trojan-based attacks extremely difficult to accomplish. A DFTT tool is also developed to automate the hardening process. The effectiveness of our Trojan prevention method is demonstrated on the Trivium encryption core.**

I. Introduction

The trend to move the IC supply chain to lower-cost locations is accelerating nowadays. Even once-trusted foundries are now vulnerable to attack, and the threat that a foundry may be compromised and malicious circuits inserted in chips it fabricates is substantial [1]. This motivates researchers to explore new testing and prevention methods, different from traditional functional and structural testing, because the characteristics of malicious circuits (i.e. hardware Trojans) are different from previously encountered anomalous behavior due to manufacturing defects or functional errors. There are several reasons why traditional testing methods are practically of no utility in detecting hardware Trojans:

1) Malicious unanticipated behavior is not included in the fault list, i.e., structural pattern testing will likely not cover Trojan test vectors [2];
2) Additional functionality of genuine designs is hard to predict without knowledge of the type of Trojan inserted by attackers;
3) Exhaustive input pattern testing is impractical as chips become more complicated with a large number of primary inputs and inner gates.

Based on these reasons, state-of-the-art EDA tools contribute little to the task of hardware Trojan detection. Only destructive reverse-engineering is potentially effective in determining whether manufactured chips are not tampered with, albeit with high testing cost. Furthermore, this method clearly can only be used on a sample group of chips with no guarantee provided that the remaining untested chips are Trojan-free [3]. Designated as "trusted IC design", researchers have to-date proposed many Trojan detection methods, which largely fall in two categories: *enhanced functional testing* and *side-channel fingerprint generation and checking* [4]. Enhanced functional testing conjectures that infrequently occurring events will be employed by attackers to trigger the hardware Trojan. Thus, the detection method is based on inclusion of these infrequently occurring events in the test plan [2], [5]. For the fingerprinting methods, the fingerprint from genuine circuits (golden models) of global power consumption [3], path delay [6], or currents on power grids [7], [8] are collected and stored. Then, these parameters are measured on every test circuit to differentiate Trojan-infected chips from genuine chips, often using advanced statistical analysis and machine learning-based methods.

However, all of the previously proposed methods are test-oriented, i.e., they do not change the design process of the IC supply chain. The authors in [4], [9], [10] have already noted the limitations of purely test-based methods by giving examples of hardware Trojans which can escape current Trojan detection methods. Thus, we turn our attention to design-stage circuit hardening in order to provide more opportunities for Trojan protection and detection.

To develop a general methodology for hardening against Register Transfer Level (RTL) Trojans, we propose a test-for-genuine procedure dealing with the original hardware description language (HDL) source code. Since the key principles of this procedure are derived from DFT (Design-for-Test), we designate our new code-hardening scheme as DFTT (Design-for-Trojan-Test). We also develop a novel DFTT tool to assist designers who have little knowledge of hardware Trojans with applying the DFTT procedure automatically.

The rest of the paper is organized as follows: Section II introduces the key principles of DFTT and the DFTT implementation process in circuit design. Section III presents our work in hardening RTL code on a Trivium-based encryption circuit [11]. Conclusions are drawn in Section IV.

II. Design for Trojan Test

A. DFTT Methodology

Our DFTT methodology inherits its concept from current DFT methods. However, for DFT methods the test vectors are generated based on the assumption that the Circuit-Under-Test (CUT) is genuine, with no inserted malicious circuits, as the purpose of DFT is to detect manufacturing faults. For our DFTT method, this would not be a valid assumption. To the contrary, we are particularly interested in generating test vectors with the objective of detecting maliciously inserted circuits whose functionality and role is unknown to us.

A key observation for DFTT methods is that despite the fact that hardware Trojans can be hidden using low-overhead, rarely-triggered circuitry, it may still be possible to detect

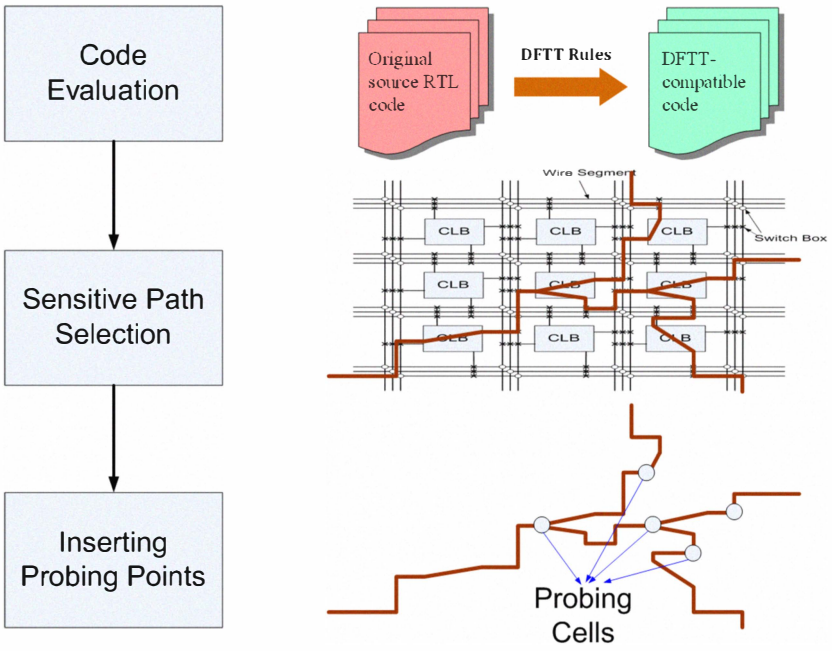

Fig. 1. The basic procedure on hardening RTL code by DFTT tool

them. However, this surely cannot be achieved with traditional functional/structural test, or even some tailored hardware Trojan detection methods [4]. Effective hardware Trojans must impose a specific *structure* on the infected circuit logic, which the attacker leverages to extract the stolen information. While this structure is not known to the defender, a circuit scrutiny of all active logic on-chip with the help of local probing cells may be sufficient to reveal its existence and, thereby, expose the hardware Trojan. Even though the attackers have a complete picture of how this scrutiny works, this method is difficult to evade. In other words, given the attacker's interest to propagate secret internal information to a primary output, our objective should be to make it exceedingly difficult to do so without touching and impacting key signal paths, which will then cause chip failure during DFTT testing.

To harden a design with our DFTT methodology, three steps need to be performed:

- **Code evaluation.** For the purpose of RTL code compliance, we develop DFTT coding rules under which effective probe cells can be inserted into user-generated RTL designs. User-generated RTL code will be converted to DFTT-compliant code in this step based on our DFTT coding rules, but circuit parameters may still be changed after the code conversion. For example, one rule requires exclusion of all glue logic in any hierarchical design level. Any top-level glue logic will be spread into lower levels. In order not to add significant overhead to the original design, several guidelines are also proposed to limit the impact of code conversion on the original design, i.e., critical paths are always prioritized during code conversion and they will be re-routed first if certain glue logic is replaced by logic in lower level modules.
- **Sensitive path selection.** The attacker's purpose is to insert an additional structure in the original design to reliably steal internal sensitive information. Randomly selected internal signals cannot always help attackers to compromise the whole chip. Since the overhead of hardware Trojans is one of the main concerns in Trojan design, attackers will try to ensure their inserted Trojans are as space-efficient as possible. Thus, attackers will first attempt to evaluate the relative merit of attacking various internal signals (such as the encryption key in a cryptographic chip) and then apply attacks upon these signals. With this consideration in mind, our DFTT tool isolates paths in which sensitive signals or other signals which are auxiliary to sensitive signals flow from primary inputs to primary outputs. Theoretically, all the internal signals are related to each other directly or indirectly. Thus, a relation degree metric is developed herein to inform the designers how closely the current signal is related to the sensitive signals. This step is critical in the whole DFTT methodology since the selection of sensitive signals is not simply a guess at the likelihood of specific hardware Trojan attacks in the nearly infinite space of potential hardware Trojan designs, but rather provides a full scrutiny of the circuit to measure the relative values of internal signals from the attacker's perspective. Inappropriate selection of sensitive paths will lead to either high circuit overhead or an unacceptable false-alarm rate.
- **Inserting probing points.** Based on the sensitive paths chosen in the previous step, probe cells are inserted into the paths of the DFTT-compliant code. This step is similar to the insertion of Scan Flip-Flops (SFF) when perform-

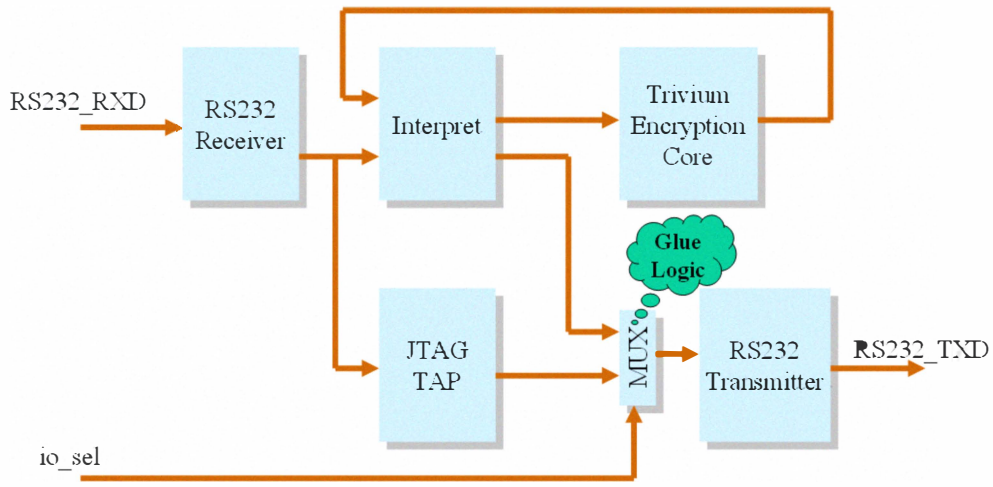

Fig. 2. The target Trivium-based encryption system

ing DFT, but the probe cell is slightly different from a normal SFF by emphasizing two key characteristics, namely *genuineness* and *integrity*. 1) The genuineness property ensures that with the help of these probing cells, testers can easily monitor internal signals and detect any abnormal behaviors triggered by additional malicious circuits. The probe cells, along with the selected sensitive paths, will ensure that any abnormal signals inside the circuit will be caught and will trigger an alarm signal to indicate a possible Trojan attack. 2) The integrity property is designed to protect probe cells themselves from attack via removal, modification, or bypassing. Clearly, only if the probe cells themselves are trusted can we trust the output of these cells. Thus, both properties of genuineness and integrity are essential to the design of probe cells.

B. DFTT Tool

Performing the three basic steps of DFTT by hand is tedious and error-prone if the target circuit is large. Therefore, in order to implement the DFTT method in large-scale circuits with limited designer involvement, we developed a design automation tool to automate the three steps mentioned above. Figure 1 shows the implementation our DFTT automation tool.

The input of the DFTT tool is RTL code of the circuit which must be hardened[1], which is converted to DFTT-compliant code in the first step. Due to the glue logic removal and design flattening, the final DFTT-compliant code, though still an RTL description, resembles more a gate-level netlist.

For the second step of sensitive path selection, the design specification will first be reviewed to extract a small set of the most valuable internal signals. These selected signals are then mapped to the DFTT-compliant code to identify all paths which these signals flow through. A specific threshold value is set here to choose paths not directly related to the selected sensitive signals. Ultimately, this threshold value will balance the area/power overhead with detection accuracy.

With DFTT-compliant code and sensitive paths available, the third step to harden a design is straightforward. The DFTT tool will read in both information and insert probe cells into the sensitive paths. Test vectors are also generated in this step. All test vectors are in the style of trigger-response pairs and can be used to test manufactured chips (or FPGA implementations).

Besides Trojan detection, another goal in designing our DFTT tool is to decrease or eliminate the probability of false alarms. That is, hardened designs are sensitive to any malicious inserted logic but should also be stable under large process variation windows and/or any circuit modification which does not change functionality. In order to achieve this goal, we add an additional code optimization stage within the first code conversion step. As long as the design specifications are the same, different Verilog code (with different coding styles) will be converted to similar DFTT-compliant code. Moreover, the hardened code output from our DFTT tool is quite robust to any subsequent modifications.

C. Trojan Detection

The final testing stage is divided into two parts: traditional testing to detect manufacturing defects and Trojan testing to detect maliciously inserted circuits. For the former stage, ATPG (Automated Test Pattern Generator) vectors are used while in the later stage, the DFT-style trigger-response pairs generated by DFTT tool are applied to the chip-under-test. In the absence of manufacturing faults, any mismatch between chip response sequence and expected corrects responses will at least expose the fact that the internal logic has been modified. Reverse engineering or other related test methods can then be implemented to further analyze the suspicious chips.

III. DFTT Implementation

To test the effectiveness of our DFTT design methodology, a Trivium-based encryption system (see Figure 2 for architectural details [12]) was ran through our DFTT tool to provide

[1]Currently our DFTT tool only supports Verilog code. We plan to extend the scope to other hardware description languages like VHDL, SystemVerilog, etc. in the future.

TABLE I
AREA OVERHEAD OF HARDENED DESIGN

	Original Design	Hardened Design	Area Overhead
Slice Flip Flops	513	580	11.6%
4-input LUT	416	961	131%

hardening. Note that the following detailed DFTT procedure is only a basic demonstration on how to implement the DFTT method on chips.

A. Code Evaluation

The DFTT tool first examined the whole design and rewrote the code where DFTT rules were violated due to the use of Trojan-vulnerable coding style. Herein, we provide one example of this code modification.

At the top module of the assignment design, a control signal `io_sel` is used to MUX the input of the UART since both ciphertext and JTAG output share one UART module. Figure 2 shows the MUX glue logic and input controlling signal `io_sel` on the top level diagram. When `io_sel='0'`, the ciphertext will be sent through the RS-232 channel. When `io_sel='1'`, the JTAG module controls the UART module. This glue logic is Trojan vulnerable and the DFTT tool relocated the corresponding gates to lower level modules. Note that in this first stage of "code evaluation", the functionality of the design was not changed.

B. Paths Selection and Probing Cell Insertion

Since the target design is a cryptographic device, signals to be protected in this design are easy to identify: the key and the plaintext from primary inputs. Two paths are isolated related to these two signals directly (since the target design is relatively small, these two paths are partly overlapping and cover most of the circuit).

Probe cells were then inserted into these two paths to increase detectability inside the circuit. In all, 480 probe cells were implemented throughout the design and 156 test vectors were generated to detect malicious modifications inside the circuit either by RTL code changes or layout modification[2].

Table I shows the area comparison in FPGA implementation between original and hardened designs of the provided cryptographic device. As we mentioned above, there is always a tradeoff between area overhead and device security in detecting/preventing hardware Trojan.

IV. CONCLUSIONS

In this paper, a new systematic Trojan prevention and RTL code hardening scheme is proposed, called Design-for-Trojan-Test (DFTT). Whereas previously proposed hardware Trojan detection methods keep the original designs unmodified, the DFTT methodology replaces Trojan-vulnerable source code with hardened Trojan prevention and detection code. Along with embedding specific probe cells, the proposed Trojan prevention and detection code will detect any malicious modification to the design. We also considered the robustness of the proposed DFTT method to ensure our method does not trigger false alarms even under substantial process variation and non-functional modifications. A DFTT tool was also developed to automate the design hardening process. Our proposed method improves chip security regardless of the chip designer's knowledge and expertise on the details of the hardware Trojan threat.

[2]In FPGA applications, the malicious modification to gate level designs can also be performed by manipulating circuit Place & Route information.

ACKNOWLEDGEMENTS

We would like to thank NYU-Polytechnic for providing us with the FPGA platform and source code of the Trivium-based encryption system. This work is partially supported by the National Science Foundation (NSF-1017719).

REFERENCES

[1] S. Adee, "The hunt for the kill switch," *IEEE Spectrum*, vol. 45, no. 5, pp. 34–39, 2008.

[2] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards Trojan-free trusted ICs: Problem analysis and detection scheme," in *IEEE Design Automation and Test in Europe*, 2008, pp. 1362–1365.

[3] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *IEEE Symposium on Security and Privacy*, 2007, pp. 296–310.

[4] Y. Jin and Y. Makris, "Hardware trojans in wireless cryptographic ics," *IEEE Design and Test of Computers*, vol. 27, pp. 26–35, 2010.

[5] H. Salmani, M. Tehranipoor, and J. Plusquellic, "New design strategy for improving hardware Trojan detection and reducing Trojan activation time," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2009, pp. 66–73.

[6] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 51–57.

[7] R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic, "Power supply signal calibration techniques for improving detection resolution to hardware Trojans," in *IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 632–639.

[8] R. Rad, J. Plusquellic, and M. Tehranipoor, "Sensitivity analysis to hardware Trojans using power supply transient signals," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 3–7.

[9] L. Lin, M. Kasper, T. Guneysu, C. Paar, and W. Burleson, "Trojan side-channels: Lightweight hardware Trojans through side-channel engineering," in *Cryptographic Hardware and Embedded Systems*, vol. 5747 of *LNCS*, pp. 382–395. Springer-Verlag Berlin, 2009.

[10] L. Lin, W. Burleson, and C. Paar, "Moles: malicious off-chip leakage enabled by side-channels," in *ICCAD '09: Proceedings of the 2009 International Conference on Computer-Aided Design*. 2009, pp. 117–122, ACM.

[11] *http://www.ecrypt.eu.org/stream/triviumpf.html.*

[12] *http://www.poly.edu/csaw-embedded.*