New Testing Procedure for Finding Insertion Sites of Stealthy Hardware Trojans

Sophie Dupuis, Bruno Rouzeyre, Marie-Lise Flottes, Giorgio Di Natale, Papa-Sidy Ba

New Testing Procedure for Finding Insertion Sites of Stealthy Hardware Trojans

Sophie Dupuis, Papa-Sidy Ba, Marie-Lise Flottes, Giorgio Di Natale, Bruno Rouzeyre
LIRMM (Université Montpellier II /CNRS UMR 5506)
Montpellier, France
{firstname.lastname}@lirmm.fr

Abstract — ***Hardware Trojans*** **(HTs) are malicious alterations to a circuit. These modifications can be inserted either during the design phase or during the fabrication process. Due to the diversity of Hardware Trojans, detecting and/or locating them are challenging tasks. Numerous approaches have been proposed to address this problem. Methods based on logic testing consist in trying to activate potential HTs and detect erroneous outputs during test. However, HTs are stealthy in nature i.e. mostly inactive unless they are triggered by a very rare condition. The activation of a HT is therefore a major challenge. In this paper, we propose a new testing procedure dedicated to identifying where a possible HT may be easily inserted and generating the test patterns that are able to excite these sites. The selection of the sites is based on the assumption that the HT (i) is triggered by signals with low controllability, (ii) combines them using gates in close proximity in the circuit's layout, and (iii) without introducing new gates in critical paths.**

I. Introduction

With ever-shrinking transistor technologies, the cost of new fabrication facilities is becoming prohibitive and outsourcing the fabrication process to low-cost locations has become a major trend in Integrated Circuits (ICs) industry in the last decade. This raises the question about untrusted foundries in which circuit descriptions can be manipulated with the possible insertion of malicious circuitry or alterations, referred to as Hardware Trojans (HTs) [1, 2, 3]. Besides, recent issues arose from the possibility of getting HTs from untrusted IP vendors also [4].

HTs detection methods are divided into two categories: methods based on *side-channel analysis* [5, 6], or *logic testing* [7, 8]. *Side channel analysis* methods focus on observing some physical parameters of the circuit, such as power consumption [5] or timing [6]. The main trend is to rely on golden ICs (i.e. circuit that have been ensured to be HT-free by destructive methods after circuit characterization) to make comparison with the circuits under test. The assumption is that the introduction of additional logic gates should become visible because of an increase of the power consumption of the circuit or an increase of the delay in the logic path containing the trigger or the payload. The main weakness of these methods is to manage environment and process variations that modify power consumption and path delays. In addition, small extra-delay introduced with trigger and payload components are difficult to detect on non-critical paths. This issue makes these methods hardly effective on small HTs.

Logic testing consists in triggering potential HTs in order to detect them during test procedure [7, 8]. HTs being stealthy in nature i.e. mostly inactive unless triggered be a rare condition, the main concern is therefore to be able to activate potential HTs i.e. to find test patterns that can maximize the chances of triggering potential HTs. The most important advantages of logic testing are that, as opposed to side channel analysis, it is robust with respect to environment and process variability and secondly does not rely on the existence of a golden circuit.

In this paper, we propose a procedure to identify circuit sites where a possible HT may be easily inserted. The selection of the sites is based on the assumption that the HT (i) is triggered by signals with low controllability, (ii) combines these signals using gates in close proximity in the circuit's layout, and (iii) without introducing new gates in critical paths. This paper is organized as follows. In Section II, we explain the context in further details. Section III presents logic testing HTs detection methods. In Section IV, we detail our technique. Experimental results are presented in Section V. Section VI discusses on the selection criteria used to identify potential HT locations. Section VII concludes the paper.

II. Context

Due to the diversity of HTs, different classifications have been proposed. The taxonomy in [7] is based on the activation mechanism (referred as the *triggering*) and the introduced effect (referred as the *payload*). The triggering logic, when it exists, monitors a set of circuit signals used to activate the payload.

Based on the assumption that the HT activation should occur under very rare conditions in order to minimize detection at test time, the HT triggering condition is supposed to depend on signals with low controllability. For the same stealthiness reasons, the payload is assumed to be attached to signals with low observability (cf. Figure 1). In this paper, we focus on this type of HTs, referred as *rare value type of triggering* based HTs in [7].

Among the different proposed HT detection methods, we focus in this paper on logic testing, i.e. on methods that rely on fault test procedures to trigger a HT and obtain an erroneous output. The aim is therefore to be able to trigger potential HTs thanks to a dedicated test sequence.

978-3-9815370-4-8/DATE15/©2015 EDAA

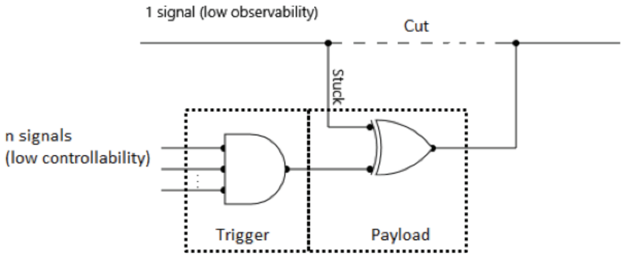

Figure 1. *Rare value* based HT circuit model [7].

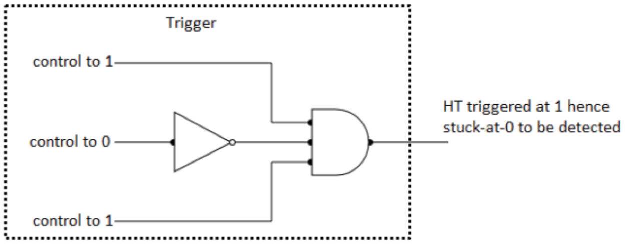

Figure 2. Test pattern generation for a particular trigger combination

The HT detection problem can be modeled as the traditional stuck-at-fault detection problem: the goal here is to control trigger inputs to the expected (low controllable) values and to observe the payload output. Concepts such as excitation, propagation, justification of expected values, typically used for generation of fault test sequences, can also be used for generation of input test sequences for triggering HTs. For example in Figure 2, to trigger the HT, we must find an input sequence that sets the trigger output to '1'. The process is similar to the one used for the generation of a test pattern for the Stuck-at-zero fault on the trigger output: (1) excitation of the trigger output to 1, (2) justification of the three trigger inputs to ('1', '0', '1'), and (3) propagation of the trigger value to the circuit output. Note that in case of a HT, the propagation of the triggering value is assumed to be done through the payload circuitry possibly introduced by an adversary. The propagation of the trigger output is then not an issue during test pattern generation for detection of potential HTs.

However, test sequence generation for HT triggering differs from the stuck-at-fault test pattern generation problem in the sense that automatic test pattern generation for fault coverage starts from a full-defined circuit netlist (e.g. gate level description), and a list of defect models (faults) precisely located on circuit's nodes. Conversely, the HT detection procedure starts without any knowledge on the HT location and implementation in the circuit. There is therefore a need for techniques dedicated to the HT detection that aim at finding a set of test patterns which maximizes the chances of triggering potential HTs while these HTs are not yet introduced in the netlist.

Possibly, Design-For-Trust solutions can be used for better detection and/or prevention as Design-For-Testability is used for testability improvement before test pattern generation.

III. Prior Work

A. Logic testing

The assumption is that a HT has a stealthy nature i.e. is activated under very rare conditions, otherwise classical production testing would most probably reveal it by accident.

Most of the time, exhaustive testing, i.e. using the exhaustive set of potential input patterns, is not an option due to the length of the test application procedure for circuits with numerous inputs and/or sequential circuits, these last ones requiring an exhaustive coverage of all combinations of primary inputs and internal states. The goal is thus to produce a reduced pattern set that maximizes the probability of activating potentials HT.

To the best of our knowledge, the first logic-based detection approach has been presented by Wolff et al. in [7]. The main idea is to find the "most likely target sites" to attach a HT trigger and stitch a HT payload. Then test patterns are generated according to this prediction. The procedure to find potential triggers sites starts with the identification of low-controllable signal sets in the circuit thanks to an exhaustive simulation. All combinations of q-signals with a low frequency of occurrence are possible triggering sets. The procedure to find potential payload sites is based on the use of a fault simulator to identify signals with a low observability. From the set of *Q* target triggers (with their corresponding trigger values) and *P* target payloads, *QxP* possible HT circuits are considered. An ATPG tool is then used to find the corresponding HT detection patterns, resulting in a compacted set of test patterns.

In [8], Chakraborty et al. propose a methodology called Multiple Excitation of Rare Occurrence (MERO). The assumption is that the number of times a HT trigger condition is satisfied increases with the number of times the trigger control signals have been individually excited to their rare value. The simulation-based procedure starts from an initial set of random patterns, a list of low-controllable signals and a user-given threshold in terms of expected number of patterns for setting low controllable signals to their rare value. For each random pattern, the procedure counts the number (*Sr)* of signals whose rare value is satisfied. The random patterns are sorted in decreasing order of *Sr*. Each pattern in the sorted list is modified with the perturbation of one bit at a time. If a modified pattern increases *Sr*, the new pattern is accepted. The procedure repeats until each signal satisfies its rare value condition for the desired number of times.

B. Design-for-trust

Design-for-trust methods aim to improve the circuit implementations in such a way that HTs become easily detectable or difficult to insert [9].

In [10], Chakraborty et al. present a methodology called *On-Demand transparency*, which aims at inserting control-points and observable points on respectively low controllable, low observable signals. An extra key input allows activating the state machine into a *transparent mode*. In this mode, the key forces the least controllable signals to their rare value. In order to improve the observation of potential HTs, the values of the least observable signals are compacted into a signature observed on an extra output.

In [11], the same authors propose to obfuscate the circuit. The method consists in modifying the state transition function of a circuit in order to create an *obfuscated mode*. The circuit starts in this obfuscated mode and only one sequence of input patterns can set the circuit to its normal mode.

Salmani et al. propose also in [12] a method that obfuscates the rare values of a circuit. In this case, the method consists in inserting so-called *dummy scan flip-flops* to facilitate the control of low-controllable nodes in test mode, reducing thus the number of potential triggering signals from the adversary point of view. Controllability of a signal is computed as probability to set the signal to '0 or '1' assuming random values on circuit's inputs.

Again with a view to obfuscate the rare values, the proposed method in [13] consists in a logic encryption of the circuit. An input key is necessary for execution in mission mode, otherwise the key changes the circuit functionality and, thanks to extra gates, changes the probabilities of occurrence of rare values on nodes identified as low controllable ones in mission mode.

C. Synthesis

Methods based on logic testing have the same goal: generating a reduced set of test patterns that is likely to trigger potential HTs. However, the two methods presented above are based on simulation results. An exhaustive simulation for trigger input identification is however very time consuming. For instance, looking for 2-input triggers in the benchmark c7522 results in 30.10^6 potential triggers.to be evaluated w.r.t exhaustive input sequences (2^{207}) or pseudo-random sequences. In the latter case, the quality of the evaluation strongly depends on the chosen set of input patterns, notably the length of the pseudo-random sequence. In addition, the procedure must be executed for several q-input triggers (q=1, 2, ….n)...

Design-for-trust method can provide assistance to the detection of HTs by increasing the controllability/observability of internal signals. They can also help prevent the insertion of a HT by virtually changing the probabilities of the signals so as to misinform an adversary who is therefore likely to insert a visible, or benign HT using easily controllable trigger and observable payload. However, it is still necessary to be able to test circuits that have not been designed following one of them.

IV. OUR HT DETECTION APPROACH

Our procedure aims at finding the most likely sites where a possible HT could be inserted as well as test patterns for detection of these potential HTs before shipping.

The criteria taken in our approach are intended to better reflect the choices that may be done by an attacker in an untrusted foundry i.e. an attacker who wants to insert a HT in a layout during the fabrication, as discreet as possible, both from the functional and from the layout point of views.

Three criteria are considered for identifying potential triggers. The first criterion is the signals' controllability. As already mentioned, signals with a low controllability, which are therefore difficult to set to a required logic value, are good candidates to create a rare triggering condition.

The second criterion is the signals' slack time (i.e. time margin). In order to hide the HT, the attacker must introduce extra gates without impacting circuit's delays; otherwise it could be discovered by HT detection methods based on delay analysis [6]. The signals with a positive slack and for which the insertion of a HT does not generate a visible degradation of delay are therefore good candidates.

The third criterion is the available space in the circuit layout. In order to hide the HT, the attacker must indeed introduce extra gates in free space not to impact the original placement [14]. Otherwise, it could be discovered by optical imaging as presented in [15]. Signals suspected to be used for HT triggering thanks to the first and the second criteria presented above are thus grouped together to form potential sets of triggering signals. Signal grouping is done according to their relative position in the layout and the available space in close proximity.

A. Controllability

The first step is to be able to find rarely activated signals in a circuit. Because exhaustive simulation is very time consuming, if not impossible, and simulation with only a portion of input patterns can lead to imprecise results, we propose to use testability analysis to find low controllable nodes.

We chose to use a *COPE-like* probabilistic testability approach as presented in [12, 16]. Our approach consists in computing the probabilities of each internal signal to be set to '0' or '1', starting from input probabilities set to ½. This method takes into account fan-out and re-convergence paths, i.e. correlations among input signals of a gate [14]. The signals with a low controllability (resp. to 0 or 1) are therefore the signals with very unbalanced probabilities to be set to '0' or '1', respectively

B. Slack time

The second step is to compute the slack time of each signal. Based on the gates delay model and the interconnect, a timing analysis is done to compute the circuit signals' slack time:

- From the set of arrival times asserted on starting points, the analysis propagates arrival times forward (As Soon As Possible),
- From the set of required arrival times asserted on end points, the analysis propagates required arrival time backward (As Late As Possible),
- The, the slack time of each signal is the difference of its required arrival time minus its arrival time.

This value is computed according to each path's delay to thwart detection methods based on delay [6]. Furthermore, the more accurate the estimation of the delay is, the better. This computation is therefore done after place and route in order to take into account not only the gates' delay, but also the interconnect's delay. Furthermore, this is what reflects the best the information that an attacker can obtain from the GDSII.

C. Subsets creation

The last step is to create subsets among the signals selected by the two previous criteria. The number of such subsets being still large, a last criterion is used to reduce it: the layout.

Signals are partitioned into subsets according to their position in the layout of the circuit. To be part of the same subset, they must be close to each other and the layout must include available space for gate insertion in their close proximity without compromising the original placement. A fourth criterion can be used for limiting the size of the signal subsets identified as potential triggering support set. Otherwise, all group sizes are searched until the distance criteria prevents from finding new subsets.

The entire procedure is described in Figure 3.

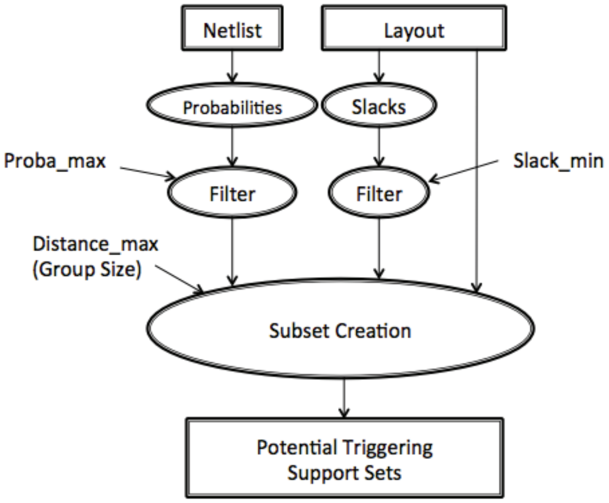

Figure 3. *Generation of potential triggering support sets*

D. Test patterns generation

Once all the subsets of potential triggering signals have been generated, the input patterns able to excite each trigger have to be found. An ATPG tool is used to obtain a test pattern for each trigger, or, if appropriate, to prove that a trigger is not justifiable.

From a practical point of view, for each triggering support set, we add to the netlist an AND gate controlled by the signals in the support set. Rare values activating the trigger when they have the value '1' are directly connected to the AND gate, while those activated for the value '0' are first inverted. The output of the AND gate is directly connected to an extra circuit primary output. The goal is here to trigger it, not to propagate it to see its effect on a primary output. We indeed assume that the trigger propagation will be done through the payload; otherwise the HT would not have any effect on the circuit behavior.

We then use the ATPG to generate a pattern to cover a stuck-at-0 at the output of that gate (such as previously described in Fig. 2). This pattern sets the logic value '1' to the AND gate output, and therefore sets the triggering signals to their rare values, the only combination that produces '1' on the AND output.

Also, since the complexity of sequential ATPG is much higher than that of combinational ATPG, we unroll the sequential parts of the circuits for several time frames to reduce the problem to an equivalent combinational version. That way we can go beyond the limitations of the tool.

V. Experimental results

We evaluated our method on two ISCAS benchmarks. First of all, we used our method to obtain potential triggers based on our three criteria. Then, we used a commercial ATPG tool [18] to obtain the corresponding set of test patterns. Last, we ran a simulation with random test patterns to see whether each trigger was indeed stealthy or not.

A. Triggers and test patterns generation

Table I presents, for each benchmark, the chosen parameters for each criterion, the number of signals meeting the first two criteria ('small' controllability and 'large' slack) and the number of triggering sets found given the last two criteria (distance in the layout and size of the triggering support set).

For each benchmark, the criteria were chosen to generate stealthy triggers:

- The probability was chosen to keep only the least controllable signals,
- The slack was chosen to remove only slacks equal to 0. The insertion of a trigger will have indeed a very little impact from a delay point of view such as detailed in [17], even a little slack would be enough to accommodate a trigger insertion, stealthy from the delay point of view.
- The distance was chosen as little as possible to be able to form groups.

With the above criteria, only 6 signals were filtered for the first benchmark, which led to 5 triggering support sets of 2 signals. For the second benchmark, 48 signals were filtered, leading to 10 2-input triggers and 1 3-input trigger.

TABLE I. Trigger sets

Bench	*Criteria*	*Number of triggering signals*	*Number of triggering sets*
c6288	Proba ≤ 0.1 Slack > 0 Distance ≤ 20 000	6	5
s13207	Proba ≤ 0.001 Slack > 0 Distance ≤ 4 000	48	11

For each potential trigger, the use of an ATPG tool allowed us to find a test pattern able to excite the trigger as explained in section IV.D.

B. *Simulation*

Table II presents simulation results for each benchmark. The percentage of times each trigger was activated is presented, during a simulation of one million random test patterns. For that purpose a tool was developed, that collects, for each test pattern of the simulation, the value of each trigger. One can notice that, since the first combinatorial benchmark has 31 inputs, one million patterns represent 0.05% of the possible input combinations. The second benchmark has 30 inputs, one million patterns represents therefore 0.1% of the possible input combinations. In addition, when dealing with a sequential circuit as the s13207 benchmark pattern order also acts upon circuit states and outputs, thus an ordered one million patterns test sequence represents a very small portion of the infinite possibilities to exercise the circuit.

According to the simulation results, four triggers appear to be not that stealthy for the first benchmark. They are activated with an average frequency of 1.34% of the input patterns. This is logical since no signal with a rare value exists in this benchmark in the first place. The six signals used (with the most unbalanced probabilities in the circuit) have a probability of 0.09 of being ‘0’, which is not that rare. The last trigger is much stealthier since it was not activated at all during the simulation (0%).

For the second benchmark, 1 triggering condition never occurs, 1 triggering condition occurs for less than 0.01% of the simulated inputs, 2 for less than 0.1%, 6 for less than 1%, and only 1 occurs for more than 1% of the patterns.

TABLE II. SIMULATION: TRIGGERS OCCURENCES

Bench	*0%*	*≤ 0.01%*	*≤ 0.1%*	*≤ 1%*	*> 1%*
c6288	1	0	0	0	4
s13207	1	1	2	6	1

VI. BEYOND THE CONTROLLABILITY CRITERIA

In all previous approaches and in our present approach, the controllability of each signal has been considered individually. It has been assumed that a triggering condition results from the simultaneous occurrence of rare values on low controllable signals. It has indeed been assumed that triggering conditions on highly controllable signals (i.e. signals with balanced probabilities to be set to ‘1’ or ‘0’) would be easily set by conventional functional or structural test sequences [7, 8, 16].

However, a *rare triggering condition* on several signals is not synonymous of a set of rare-value on low controllable signals. For instance, two signals may have a 0.5 probability of being ‘0’ (full controllability to ‘1’ or ‘0’) while the combination (0,0) never arises. In other words, it is reasonable to assume that a rare triggering condition can also be composed of a set of signals that do not have individually a low controllability. We therefore aimed at studying this variant in the trigger definition.

We therefore developed a second procedure in which only the slack and the position in the layout of the signals are considered to create the triggering subsets. Then, the probabilities of each subset value are computed to filter rare values. To do so, a variant of our procedure is described in Figure 4.

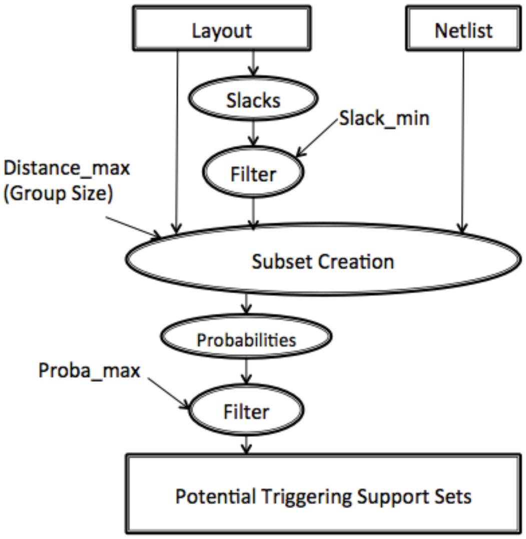

Figure 3. *Our proposed scheme (2).*

A. *Triggers and test patterns generation*

We evaluated this new method with the same two benchmarks. Results are presented in Table III. The selecting criteria are presented for each benchmark. For the first benchmark, among the 274 triggering sets found, 54 have a probability under 0.05. However, 25 have a probability of 0, which leaves 29 potential triggering sets. For the second benchmark, among the 159 triggering sets, 46 have a probability under 0.001 (25 equal to 0), which produces 21 potential triggers.

For the triggers found, we have analyzed the probabilities of the signals composing the triggers, such as presented in Table IV. For the first benchmark, among the 29 triggering sets, only 4 contain a signal with unbalanced probabilities (low controllability to ‘0’ or ‘1’). For the second benchmark, among the 21 triggering sets, only 2 are composed of 2 signals with unbalanced probabilities, 17 have 1 signal out of 2 with unbalanced probabilities (and the other one with probabilities up to 0.5/0.5), and 2 are composed of signals without unbalanced probabilities.

Then, the use of a commercial ATPG tool has allowed us to find a test pattern being able to excite each trigger.

B. *Simulation*

Table V presents simulation results for each benchmark. The percentage of times each trigger was activated is presented, during a simulation of one million random test patterns. The simulations confirm the rareness of the triggering conditions. Five triggering conditions never occur for the first benchmark and 3 for the second benchmark, while 3 others triggering conditions appear for less than 0.01% of the simulated data.

These results raise serious doubts about the selection of triggering conditions as proposed in the literature up to now. The issue of a stealthy trigger composed of signals that do not have a individual low controllability measure is an alternative that deserves to be studied in more detail in the future.

TABLE III. TRIGGER SETS (2)

Bench	*Criteria*	*Number of triggering sets*
c6288	Slack>0	
	Distance≤10000	
	Group size=2	274
	Probability≤0.05 & >0	29
s13207	Slack>0	
	Distance≤2000	
	Group size=2	159
	Probability≤0.001 & >0	21

TABLE IV. TRIGGERS COMPOSITION

Bench	*Rare / Rare*	*Rare / Non rare*	*Non rare / Non rare*
c6288	0	4	25
s13207	2	17	2

TABLE V. SIMULATION: TRIGGERS OCCURENCES (2)

Bench	*0%*	*≤ 0.01%*	*≤ 0.1%*	*≤ 1%*	*> 1%*
c6288	5	0	0	0	24
s13207	3	3	10	4	1

VII. CONCLUSION

In this paper, we have presented a procedure to identify circuit sites where a possible HT trigger may be easily inserted in an untrusted foundry. The selection of the sites is based on the assumption that the HT is triggered (i) by signals with low controllability, (ii) in paths that are not critical in terms of delay, and (iii) combining multiple signals that are close one to the other in the circuit's layout, and close to available space.

The computation of the controllability of the signals is not based on simulation results, which can lead to imprecise results, based on the test sequence chosen. From the found potential triggers, the corresponding test patterns are determined using an ATPG tool.

Furthermore, we have studied a variant in the commonly used definition of triggers, to establish that a so-called *rare value* could be composed of signals that do not necessarily have a low controllability.

ACKNOWLEDGMENTS

This project has been funded by the French Government (BPI-OSEO) under grant FUI#14 **HOMERE** (**H**ardware tr**O**jans : **M**enaces et robust**E**sse des ci**R**cuits int**E**grés).

REFERENCES

[1] X. Wang, M. Tehranipoor and J. Plusquellic. Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08)*, pp.15–19, 2008.

[2] M. Tehranipoor and F. Koushanfar. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Design & Test of Computer,* 27:10–25, 2010.

[3] S. Bhunia, M. S. Hsiao, M. Banga, S. Narasimhan. Hardware Trojan Attacks: Threat Analysis and Countermeasures. In *Proceedings of the IEEE, Special Issue on Trustworthy Hardware*, 102(8):1229–1247, 2014.

[4] Y. Jin and Y. Makris. Proof Carrying-Based Information Flow Tracking for Data Secrecy Protection and Hardware Trust. In *IEEE VLSI Test Symposium (VTS'12)*, pp. 252–257, 2012.

[5] D.Agrawal, S.Baktir, D.Karakoyunlu, P.Rohatgi, and B.Sunar. Trojan Detection using IC Fingerprinting. In *IEEE Symposium on Security and Privacy (SP'07),* pp. 296–310, 2007.

[6] Y. Jin and Y. Makris. Hardware Trojan Detection Using Path Delay Fingerprint. In *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08),* pp. 51–57, 2008.

[7] F. Wolf, C. Papachristou, S. Bhunia and R. S. Chakraborty. Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme. In *Design, Automation and Test in Europe (DATE'08)*, pp. 1362–1365, 2008.

[8] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia. MERO: A Statistical Approach for Hardware Trojan Detection. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES'09),* pp. 396–410, 2009.

[9] J. Rajendran, O. Sinanoglu, R. Karri. Regaining Trust in VLSI Design: Design-for-Trust Techniques. In *Proceedings of the IEEE, Special Issue on Trustworthy Hardware.* 102(8):1266–1282, 2014.

[10] R. S. Chakraborty, S. Paul, S. Bhunia. On-Demand Transparency for Improving Hardware Trojan Detectability. In IEEE *International Workshop on Hardware-Oriented Secutity and Trust (HOST'08)*, pp. 48–50, 2008.

[11] R. S. Chakraborty, S. Bhunia. Security against Hardware Trojan through a Novel Application of Design Obfuscation. In *International Conference on Computer-Aided Design (ICCAD'09)*, pp. 113–116, 2009.

[12] H. Salmani, M. Tehranipoor, and J. Plusquellic. A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* 20(1):112–125, 2012.

[13] S. Dupuis P.-S Ba, G. Di Natale, M.-L. Flottes, B. Rouzeyre. A Novel Hardware Logic Encryption Technique for thwarting Illegal Overproduction and Hardware Trojans. In *International On-Line Testing Symposium (IOLTS'14)*, pp. 49–54, 2014.

[14] K. Xiao, M. Tehranipoor. BISA: Built-In Self-Authentication for Preventing Hardware Trojan Insertion. In *International Symposium on Hardware-Oriented Security and Trust (HOST'13), pp.* 45–50, 2013.

[15] S. Bhasin, J.-L. Danger, S. Guilley, X. T. Ngo, L. Sauvage. Hardware Trojan Horses in Cryptographic IP Cores. In *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'13), pp.15–29, 2013.*

[16] G. Di Natale, S. Dupuis, M.-L. Flottes, R. Rouzeyre. Identification of Hardware Trojans triggering signals. In *Workshop on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE'13)*, 2013.

[17] S. Dupuis G. Di Natale, M.-L. Flottes, B. Rouzeyre. On the Effectiveness of Hardware Trojan Horse Detection via Side-Channel Analysis. In *Information Security Journal: A Global Perspective*, 22:226–236, 2013.

[18] Synopsys Tetramax, http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Test/Pages/TetraMAXATPG.aspx