Using Outliers to Detect Stealthy Hardware Trojan Triggering?

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

Using Outliers to Detect Stealthy Hardware Trojan Triggering?

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

Abstract— **Hardware Trojans (HTs) are malicious alterations to a circuit introduced at design or manufacturing phases by an adversary. Due to their diversity, detecting and/or locating them are challenging tasks. Among the different kinds of detection methods, methods based on logic testing aim to reveal the presence of HTs thanks to their logical activation and observation through primary outputs. However, HTs are stealthy in nature, i.e. mostly inactive unless triggered by a very rare condition. Furthermore, test patterns must be developed without any knowledge on the location and function implemented by the HT.**

A procedure has recently been proposed to identify in a netlist suspicious signals that may be part of a HT introduced at design stage: these so-called "outliers" present an activity poorly correlated with other signals. In this paper, we undertake to use outliers in order to detect a HT introduced at manufacturing stage. We propose a test pattern generation technique based on the exploration of outliers. Our assumption is that such signals may be selected by an attacker in order to trigger his/her HT.

Keywords- Hardware Trojan Detection; Logic testing; Cross correlation; Clustering; Outliers

I. INTRODUCTION

With ever-shrinking transistor technologies, the cost of new fabrication facilities is becoming prohibitive. Outsourcing the fabrication process to low-cost locations has therefore 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, 5].

HTs detection methods on fabricated ICs are commonly divided into two categories: methods based on *side-channel analysis* [6-8], or *logic testing* [9-11]. In both cases the goal is to test ICs after fabrication to ensure that they are HT free. In addition, Design-for-Hardware-Trust methods aim at helping HT detection methods or preventing an attacker from inserting HT thanks to specific design rules [12]. These methods can for example modify the combinational logic [13, 14].

Side channel analysis methods focus on observing some physical parameters of the circuit, such as power consumption [6, 7] or timing [8]. The principle is to rely on golden ICs, i.e. circuits 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 gates should become visible because of an increase of the power consumption or the path delay in the logic path containing the trigger or the payload.

Logic testing methods consist in activating potential HTs in order to provoke an error on circuit's outputs [9-11]. A HT model is presented in [9], decomposing the HT into to parts: the trigger and the payload (cf. Fig 1). The assumption is that the HT is stealthy i.e. dormant most of the time, until activated by the trigger, under a very rare condition, so that the payload propagates the harmful effect. The main concern is therefore to be able to trigger 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, it does not rely on the use of a golden circuit.

Besides, searching for HTs at design stage has been studied recently [15-18]. Based on the assumption that a HT can be incorporated in an IP or even in the design by a rogue designer, the idea is to identify in a netlist "suspicious logic" that may be part of a HT.

In this paper, we propose to use the notion of "outliers", i.e., signals that do not have an activity similar to the general distribution, to identify signals that may be interesting for attacker in an untrusted foundry to create a HT. They seem indeed suitable to create so-called "rare events" that should escape conventional test. This paper is organized as follows. In Section II we recall different detection methods using logic testing as well as methods searching for HTs at netlist level. In Section III, we detail our technique. Experimental results are presented in Section IV. A conclusion and future works are presented in Section V.

978-1-5090-1141-4/16/$31.00 ©2016 IEEE

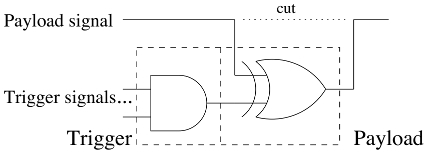

Figure 1. HT model

II. Background

A. Logic testing

The goal of every detection method is to trigger potential HTs in order to provoke an error observable on circuit's outputs. The common assumption is that HTs are necessarily stealthy in order to evade detection during conventional test procedures using random and/or ATPG patterns. The main goal for detection methods is then: (1) to find the "most likely signals" on which an attacker could have attached a HT trigger and/or stitched a HT payload, (2) to find the corresponding HT detection patterns (using an ATPG tool). These procedures result in a reduced set of patterns that is likely to activate potential triggers and propagate potential payloads.

To the best of our knowledge, the idea of using logic testing in order to detect HTs has been introduced by Wolff et al. in [9]. The assumption is that the inputs of a HT trigger depend on the least controllable signals of a circuit, and payloads on the least observable signals. The procedure to find potential triggers sites starts with the identification of low-controllable signals thanks to, if possible, an exhaustive simulation. Most of the time however, an exhaustive approach will not be affordable due to the large number of inputs. A random pattern simulation has to be performed instead. All combinations of signals with a low frequency of occurrence are considered as potential 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 to generate the suitable set of patterns.

The same authors improve the patterns generation step in [10], in which they propose a methodology called Multiple Excitation of Rare Occurrence (MERO). The idea is to propose a set of patterns that activates each potential trigger a given number of times: the assumption is that, similarly to N-detect test, triggering each of the signals individually multiple times should increase the probability of triggering (and thus detecting) potential HTs.

Another logic testing approach is presented in [11]. The goal is also to identify potential trigger inputs and generate efficient patterns to activate these triggers. However, conversely to previous works, low controllable signals are identified thanks to controllability metrics (based on equally probable input signals) instead of simulation data. The assumption is that, since exhaustive simulation is unaffordable and random patterns could leave to imprecise results, controllability metrics should be more accurate. Furthermore, several criteria are considered in [11] to identify potential triggers, assuming that the attacker's goal is to insert a HT as stealthy as possible, from the functional, performance, and layout point of views. The selection of the potential triggers is therefore 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 from each other in the circuit's layout.

B. Detection at design stage

A method called "Unused Circuit Identification" (UCI) is presented in [15]. In order to identify in a netlist "suspicious logic" that may be part of a HT, the idea is to detect so-called *unused logic* i.e. logic activated by any of the design verification tests. UCI generates a list of all signal pairs and does a simulation to find pairs that are equal throughout all tests. However, a successful implementation of a HT defeating UCI was proposed shortly after in [19], showing a HT without any pair of signals that are always equal when the HT is not activated.

In [16], the method "VeriTrust" aims to identify HTs trigger inputs by examining verification corners, under the assumption that HTs trigger inputs are usually not sensitized with verification test cases. In other words, the method looks for redundant inputs, after setting all un-activated entries to "don't cares" during test. Firstly, a tracer traces verification tests to identify signals that contain un-activated entries. Secondly a checker analyzes these signals to determine if they contain redundant inputs and are therefore potentially affected by a HT. As opposed to [15], VeriTrust is insensitive to HT implementation styles.

The assumption presented in [17] is that the logic belonging to a HT almost never influences the outputs of the circuit. The concept of "control value" is introduced, which describes how signals affect other signals. The goal is then to identify the signals that have an abnormally low degree of influence. The computation of the control values is done by approximating the truth table for each signal i.e. by randomly sampling rows in the truth table.

The approach presented in [18] estimates the statistical correlation between signals. Using simulation data aiming at exciting as many nodes as possible, a correlation-based similarity weight is computed for the input-output pair of each gate in a circuit. Then a clustering algorithm is used to detect so-called outliers, by creating a clustering structure and detecting signals that are pushed with high-reachability distances to the border of the clusters. The insight is that a HT has weak statistical correlation with the rest of the circuit.

III. Proposed approach

The underlying principle of the proposed work is the same as in every detection method based on logic testing: finding potential HT trigger inputs and generating the corresponding test vectors.

To find trigger inputs, the insight of the proposed work is to focus on outliers, as introduced in [18]: the principle is then to run a simulation, to analyze the frequency behavior of each signal and to determine outliers i.e. signals that are not correlated to the rest of the signals.

Note that, unlike the work [18], we do not seek a HT inserted at design stage, but HT inserted during fabrication. In other words, we do not seek for signals inside a HT, but rather signals that may be used to attach a HT trigger. Our goal is to show whether or not this criterion is applicable in this case.

A. *Information theory*

The first step is to compute the "energy" between the inputs and the outputs of each gate. It allows to obtain the cross correlation between every input/output pair.

Mathematically, the energy is the module of the Discrete Fourier Transform (DFT). The first step is then to compute the convolution.

1) Convolution:

An Automatic Test Pattern Generation (ATPG) tool is used to generate input patterns. Then, a simulation is run to obtain the responses and create a vector for each signal given its successive values.

Let's consider for example that a signal a takes, during simulation, the following values:

v(a) = {10101010}

From this vector, the switching vector of signal a is determined as follows:

switch(a)[0] = v(a)[0]

switch(a)[i] = v(a)[i] xor v(a)[i-1]

Then, the convolution is computed given the switching vector of each signal. Let's consider two signals a and b driving a two-input gate, signal g being the output of that gate and SV_a (resp. SV_b, SV_g) being the switching vector of a (resp. b, g). The convolution c_{ag} between a and g is stated as follows:

$$c_{ag}[n] = (SV_a * SV_g)[n] = \sum_{m=-\infty}^{+\infty} SV_a[m] SV_g[m+n] \qquad (1)$$

2) Discrete Fourier Transform:

The DFT of convolution between a and g is as follows:

$$S_{ag}(k) = \sum_{n=0}^{N-1} c_{ag}(n).e^{-2i\pi k\frac{n}{N}} \qquad (2)$$

3) Energy:

Then, the energy between a and g is as follows:

$$\Psi_{S_{ag}}(k) = \left|s_{ag}(k)\right|^2 \qquad (3)$$

$$\psi_{S_{ag}}(k) = Parseval\,identity \qquad (4)$$

The energy gives information about the frequency with a minimal number of coefficients.

Since we look for the dissimilarity between signals according to their activity (commutation), the energy will give us the similarity between neighboring nodes.

B. *Clustering*

Our objective is to find, from a given dataset, isolated points considered as outliers. Given these signals' feature, they may be used as triggers for HTs. To do so, we use a clustering algorithm based on a distance matrix. Clustering is indeed the task of grouping a set of objects in such a way that objects in the same group are "similar to each other". The clustering algorithm will then group into clusters signals with a similar activity and signals that won't belong to any cluster will be considered as outliers.

1) Energy normalization into a distance matrix

First, the netlist is transformed into a weighted graph (energy = weight) and the weight is normalized. Then, the structural similarity σ(u,v) between two adjacent signals u and v is determined as follows [20]:

$$\sigma(u,v) = \frac{\sum_{x\in\Gamma(u)\cap\Gamma(v)} w(u,x).w(v,x)}{\sqrt{\sum_{x\in\Gamma(u)} w^2(u,x)}\sqrt{\sum_{x\in\Gamma(v)} w^2(v,x)}} \qquad (5)$$

where Γ(u) denotes the neighborhood of signal u, including u and all its adjacent signals (w(u,u) is defined as 1).

Given the similarity values of every signal, the distance between each signal and its adjacent signals is defined to be inversely proportional to the corresponding similarity rate.

The distance between non-adjacent signals u and v is defined as the sum of the distances on the shortest path that connects them. We use Dijkstra algorithm to compute such paths:

$$\text{distance}(u,v) = \frac{1}{\sigma(u,v)} \qquad (6)$$

2) Transformation of the distance matrix (N*N) into a coordinates matrix (N*2)

Our distance matrix is multidimensional (N*N). A transformation is performed in order to obtain a coordinates matrix (N*2). With those coordinates, we will be able to obtain a representation of all signals according to their activity (switching). To perform that transformation, we use the Classical Multidimensional Scaling method in [22].

3) Outlier detection algorithm

In [23], a method to find outliers is presented. Outliers are objects deviating from the major distribution of the data set. In other words, being an outlier means not being in or close to any cluster. Using cross correlation, the outliers will be the signals that are not correlated to the others. This algorithm is a density-based clustering structure. The outlier factor is defined as follows:

$$of_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPis}(p)|} \quad \textbf{(7)}$$

where lrd denotes the local reachability density of an object p to be the inverse of the average reachability-distance from the MinPts-nearest-neighbors of p (MinPts is the minimum number of points within the neighborhood of p).

The outlier factor of object p captures the degree to which p can be considered as an outlier. It is the average of the ratios of the local reachability density of the MinPts-nearest-neighbors and p. If these are identical, which we expect for objects in clusters of uniform density, the outlier is 1.

4) Outlier factor normalization

In order to facilitate selection, all outliers' factor values should be between 0 and 1. To do so, we use the method presented in [24], in which a Gaussian scaling is used to normalize outlier factors. According to the central limit theorem, the most general distribution for values derived from a large number of similarity-distributed values is the normal distribution. Having just two degrees of freedom (mean μ and standard deviation σ). Giving the mean and the standard deviation of the set of derived values using the outlier factor, we can use its cumulative distribution function and the Gaussian error function erf() to transform the outlier factor into a probability P_o. It is defined as follows:

$$\begin{aligned}
&of \sim N(\mu, \sigma) \\
&\mu = E(of) \\
&\sigma^2 = E(of^2) - E(of)^2 \\
&P_o = \frac{1}{2}(1 + erf\left[\frac{of_a - \mu}{\sigma * \sqrt{2}}\right])
\end{aligned} \quad \textbf{(8)}$$

C. Triggers identification

Once outliers obtained, the idea is to create combinations of these outliers to form potential triggers, and then to generate test patterns activating them using an ATPG tool. Note that if the generation of a pattern activating a potential trigger is impossible, the trigger is ignored, since non-triggerable.

Note that, assuming outliers are good candidate from a stealth point of view, the notion of outlier can also be combined with the delay information and/or the layout position as introduced in [11] to from the triggering sets.

IV. EXPERIENCES

Our method was evaluated on different ISCAS benchmarks. We used Synopsys tools for synthesis and Pattern generation and a 65 nm standard cells library.

A. Outliers on HT free circuits?

We applied our method on HT-free circuits to determine if outliers would be found or not. After running a simulation with random patterns, we ran the outlier detection algorithm and got a coefficient (outlier factor) for each signal. We then normalized the coefficients into probabilities (from 0 to 1).

Table I presents the numbers of outliers found in each benchmark (i.e. signals with probabilities equal to 1), and signals that could be outliers with a lower probability threshold: greater than 0.98 and greater that 0.95. Obviously, the less constrained the probability is, the more outliers are detected. It's interesting to notice that outliers are found in every benchmark. That proves that searching for outliers in HT-free circuits may be interesting and we ought to find how to use this criterion properly in that case. Besides, we ran three different simulations on one benchmark to evaluate the impact of the patterns on the results. The same outliers were identified in all cases.

Figures 2 to 4 show the results we obtained for three of the benchmarks after transforming the distance matrix into 2-d dataset. In Fig. 2 and 3, a cluster seems to appear, but not very dense. In Fig. 2, one signal is clearly outside that cluster (the one to the right), which corresponds to the outlier found in Table I. In Fig. 3, several signals do not belong to the main cluster, corresponding to the 9 outliers found in Table I. In Fig. 4, a very dense clustered is displayed, and 4 signals do not belong to that cluster, corresponding to the 4 outliers in Table I.

TABLE I. NUMBER OF OUTLIERS

Benchmark	Number of signals	Probability for being an outlier (P_o)	Number of outliers
C432	196	$P_o = 1$	1
		$0.98 \leq P_o \leq 1$	7
		$0.95 \leq P_o \leq 1$	14
C1355	587	$P_o = 1$	17
		$0.98 \leq P_o \leq 1$	51
		$0.95 \leq P_o \leq 1$	60
C3540	1719	$P_o = 1$	9
		$0.98 \leq P_o \leq 1$	105
		$0.95 \leq P_o \leq 1$	115
C6288	739	$P_o = 1$	4
		$0.98 \leq P_o \leq 1$	4
		$0.95 \leq P_o \leq 1$	4
S13207	877	$P_o = 1$	9
		$0.98 \leq P_o \leq 1$	20
		$0.95 \leq P_o \leq 1$	20
S35932	5677	$P_o = 1$	51
		$0.98 \leq P_o \leq 1$	70
		$0.95 \leq P_o \leq 1$	70

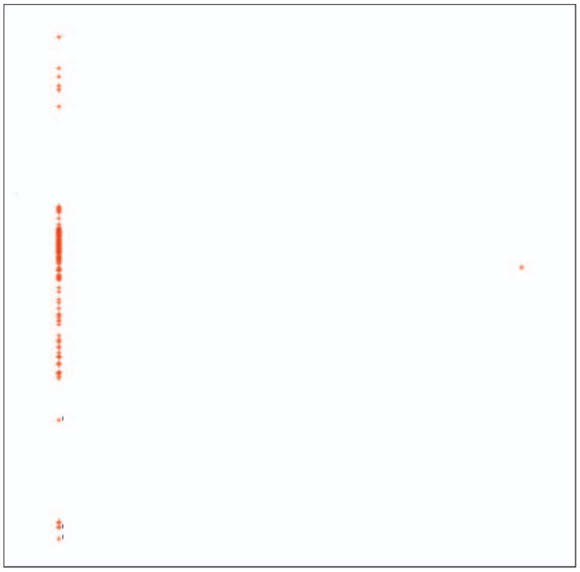

Figure 2. 2-d dataset of C432

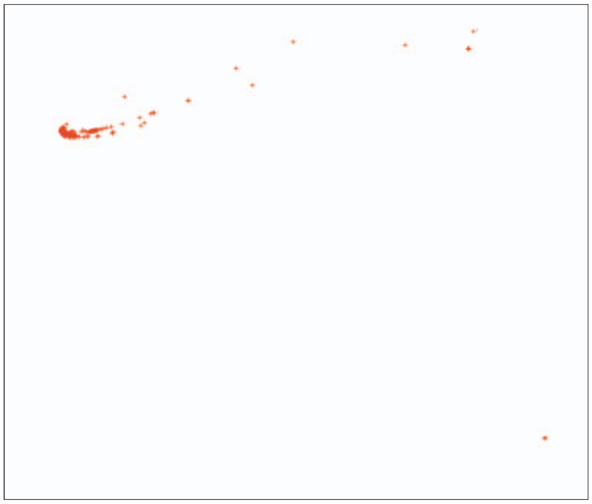

Figure 3. 2-d dataset of C3540

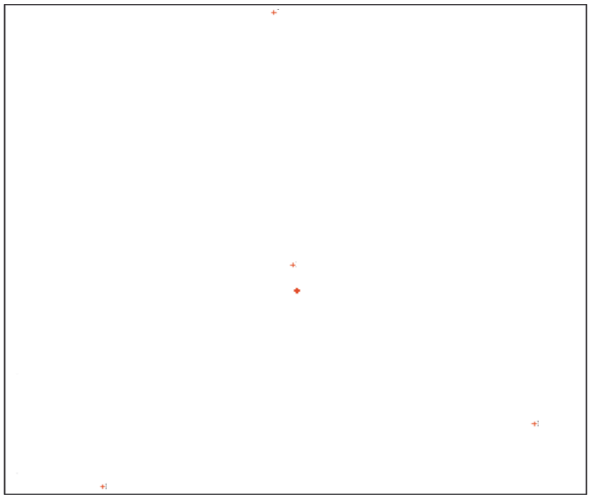

Figure 4. 2-d dataset of C6288

From these data, one can conclude that, in these HT free circuits, several nodes do not have a similar activity to the general group and are therefore considered as outliers. A question remains: can these outliers actually be interesting for an attacker in an untrusted foundry wishing to insert a HT?

B. *Outliers as triggers?*

To prove whether or not outliers can be used by an attacker in order to find potential HT triggers, we inspected the activity of each outlier during a simulation.

Table II presents the behavior of the four signals denoted as outlier in benchmark c6288 during a simulation of one million random patterns. It turns out that the outliers found are signals that nearly never commute (from 1 to 18 times).

These data show that our initial assumption is correct: outliers found in HT free circuits can be used to create a stealthy trigger.

V. CONCLUSION

In this paper, we investigated if “outliers” were interesting for an attacker in an untrusted foundry. In other words, we investigated if such signals that are not correlated to the general switching activity could be used to create a HT’s trigger. The notion of outlier was introduced in the literature to detect HTs introduced during design. We assumed that it could also be used to detect potential triggers inputs and aimed at proving that theory.

From experimental results, we have shown that outliers can be found in HT free circuits and that their switching activity makes them interesting for an attacker. However, this concept does not provide more information than previous works [9-11]: it is a new way to get the same information, low-controllable signals.

As future work, we plan to extend the presented work by investigating signals that are not identified as outliers: the idea is to check if combinations of signals that are not necessarily outliers, but that are as far as possible from each other in a cluster, or that belong to different clusters, allow to create a stealthy condition. The idea of triggers composed of signals that are not individually low controllable was introduced in [11] and deserves to be investigated.

TABLE II. C6288 OUTLIERS’ BEHAVIOR (SIMULATION OF ONE MILLION PATTERNS)

Number of ‘0’	Number of ‘1’
999982	18
999991	9
999999	1
999999	1

Acknowledgment

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”. In IEEE Design & Test of Computer, 27:10– 25, 2010.

[3] S. Bhunia, M. S. Hsiao, M. Banga and 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] X. Zhang and M. Tehranipoor. “Case Study: Detecting Hardware Trojans in Third-Party Digital IP Cores”. In IEEE International Symposium on Oriented Security and Trust (HOST’11), pp. 67–70, 2011.

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

[7] X. Wang, H. Salmani, M. Tehranipoor and J. Plusquellic. “Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis”. In IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems (DFT’08), pp. 87–95, 2008.

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

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

[10] 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.

[11] S. Dupuis, P.-S. Ba, M.-L. Flottes, G. Di Natale and B. Rouzeyre. “New Testing Procedure for Finding Insertion Sites of Stealthy Hardware Trojans”. In Design Automation & Test in Europe (DATE’15), pp. 776–781, 2015.

[12] J. Rajendran, O. Sinanoglu and 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.

[13] H. Salmani, M. Hassan 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 (HOST’09), pp. 66–73, 2009.

[14] S. Dupuis, P.-S. Ba, G. Di Natale, M.-L. Flottes and 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.

[15] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith. “Overcoming an Untrusted computing base: detecting and removing malicious hardware automatically”. In Symposium on Security and Privacy (SSP’10), pp.159–172, 2010.

[16] J. Zhang, F. Yuan, L. Wei, Z. Sun and Q. Xu. “Veritrust: Verification for Hardware Trust”. In Design Automation Conference (DAC’13), 2013.

[17] A. Waksman, M. Suozzo and S. Sethumadhavan. “FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis”. In ACM Conference on Computer & Communications Security (CCS’13), pp. 697–708, 2013.

[18] B. Cakir and S. Malik. “Hardware Trojan detection for Gate-level ICs using signal correlation Based Clustering”. In Design, Automation & Test in Europe (DATE’15), pp. 471–476, 2015.

[19] C. Sturton, M. Hicks, D. Wagner and S. T. King. “Defeating UCI: Building Steakthy and Malicious Hardware”. In Symposium on Security and Privacy (SSP’11), pp. 64–77, 2011.

[20] J. Huang, H. Sun, J. Han, H. Deng, Y. Sun and Y. Liu. “SHRINK: A Structural Clustering Algorithm for Detecting Hierarchical Communities in Networks”. In International Conference on Information and Knowledge Management, pp. 121–132, 2010.

[21] M. Ankerst, M. M. Breunig, H.-P Kriegel and J. Sander. “Ordering Point To Identify the Clustering Structure”. In International Conference on Management of Data, pp. 49–60, 1999.

[22] J. Wang. “Classical multidimensional scaling”. In Geometric Structure of High-Dimensional Data and Dimensionality Reduction. Springer Berlin Heidelberg, pp. 115–129, 2011.

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng and J. Sander. “OPTICS-OF: Identifying Local Outliers”. In European Conference in Principles of Data Mining and Knowledge Discovery, pp. 262–270, 1999.

[24] H.-P. Kriegel, P. Kroger, E. Schubert and A . Zimek. “Interpreting and unifying outlier scores”. In SIAM International Conference on Data Mining, 2011.