A Novel Hardware Logic Encryption Technique for thwarting Illegal Overproduction and Hardware Trojans

Sophie Dupuis, Papa-Sidi Ba, Giorgio Di Natale, Marie-Lise Flottes, Bruno Rouzeyre
LIRMM (Université Montpellier II /CNRS UMR 5506)
Montpellier, France
{dupuis, ba, dinatale, flottes, rouzeyre}@lirmm.fr

Abstract — **Hardware piracy is a threat that is becoming more and more serious these last years. The different types of threats include mask theft, illegal overproduction, as well as the insertion of malicious alterations to a circuit, referred to as Hardware Trojans. To protect circuits from overproduction, circuits can be encrypted so that only authorized users can use the circuits. In this paper, we propose an encryption technique that also helps thwarting Hardware Trojan insertion. Assuming that an attacker will attach a Hardware Trojan to signals with low controllability in order to make it stealthy, the principle of the encryption is to minimize the number of signals with low controllability.**

Index Terms—**Hardware Trojan; Hardware Trojan Detection; Hardware Trojan Activation; Logic Encryption; Logic testing**.

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 and therefore hardware piracy: mask theft and unauthorized overproduction [1, 2] as well as the insertion of malicious circuitry or alterations, referred to as Hardware Trojans (HTs) [3, 4]. Since the fabrication process becomes untrusted, IC vendors face two challenges: protect the ICs from mask theft and be able to verify the trustworthiness of the manufactured ICs.

One way to protect the ICs from mask theft and illegal overproduction is to encrypt the design so that only authorized users can use the circuits. This is achieved by hiding the functionality of a design with the addition of a key into the design that is necessary for the proper operation of the circuit: the circuit produces correct outputs only if the valid key is applied.

Logic encryption can be classified into two types: combinational encryption (i.e. modification of the gate level netlist [1, 2]) and sequential encryption (i.e. modification of the state transition graph [5]). In the first case, additional logic gates are inserted to conceal the functionality of the design upon applying a wrong key. In the latter case, the state transition is modified so that the design reaches a valid state only by applying a correct sequence of inputs.

Due to the diversity of HTs, detecting them is a challenging task. Different classifications of HTs have therefore been proposed. The proposed classification in [6] is based on the activation mechanism (referred as the *triggering*) and the introduced effect (referred as the *payload*). The triggering logic monitors a set of signals to activate the payload at the proper event. The fundamental assumption in that case is that the HT activation should occur under very rare conditions i.e. the trigger is attached on signals with low controllability and the payload is attached on signals with low observability. This is referred in [6] as *rare values* based HTs. A model of this type of HT is presented in Figure 1.

HTs detection methods are divided into two categories: methods based on *side-channel analysis* [7, 8], or *logic testing* [6, 9]. In the latter case, logic testing is performed before deployment in the field to try to detect the presence of potential HTs: if an erroneous behavior of the IC is observed, it can be inferred that a HT has been inserted in the IC. Yet, due to the stealthiness of the HTs, traditional ATPG test vectors are not sufficient to detect them. The main concern is therefore to be able to activate potential HTs i.e. to find test vectors that can maximize the chances of activating potential HTs [6].

Furthermore, so-called *design for hardware trust* methods can also be used. They consist in incorporating into the ICs some features that should improve the HT detectability [5, 10].

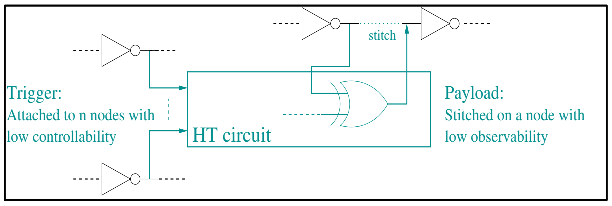
stitch
Trigger:
Attached to n nodes with low controllability
HT circuit
Payload:
Stitched on a node with low observability

Figure 1. *Rare value* HT circuit model in [6].

In this paper, we propose a combinational encryption technique that allows protecting ICs against illegal overproduction and helps thwarting the insertion of potential HTs by minimizing *rare values* in the circuit i.e. minimizing the number of signals with low controllability. This way, it makes it harder for an attacker to exploit the true rare values in a circuit to incorporate a HT. These modifications do not change the functionality of the design. Moreover, they are done in a way such that they do not represent a large additional cost in area and do not alter the timing either.

This paper is organized as follows. In Section II, we recall the different proposed HTs detection methods based on logic testing as well as encryption methods. In Section III, we present our encryption technique. In Section IV, we present some simulation results. Finally, Section V concludes the paper.

II. PRIOR WORK

A. *Logic encryption*

The goal of logic encryption is to protect ICs from mask theft and unauthorized production. The principle is to prevent unauthorized users to use the ICs. This requires every IC to be activated with an external key.

A combinational encryption technique is proposed in [1]. It consists in randomly inserting XOR/XNOR gates into the design. One input of each of these gates is connected to a newly added key input.

A second approach, presented in [2], improves this technique by ensuring that wrong keys corrupt the outputs. The places to insert the XOR/XNOR gates are chosen in order to achieve 50% Hamming distance between the correct and the wrong outputs.

In [5], a *design obfuscation* scheme is presented which modifies the state transition functions in order to create two functional modes: the normal mode and the obfuscated mode. By default, the IC is in obfuscated mode and the key (a specific input sequence) allows toggling from obfuscated mode to normal mode. Not only this approach helps fighting against overproduction, but it also helps thwarting HT insertion by e.g. making some inserted HTs benign (those effective in obfuscated mode).

B. *HTs detection methods based on HT activation*

In order to be able to detect a potential HT by logic testing, the main concern is to be able to activate the HT. The assumption is that a HT has a stealthy nature i.e. is activated under very rare conditions. Hence, the HT detection methods aim at optimizing pattern generation techniques to maximize the probability of inserted HTs getting activated and therefore detected by logic testing.

The first logic based detection approach is presented in [6]. The goal is to find so-called *HT test vectors* (i.e. vectors that can detect HTs triggered by rare values). It is assumed that HTs triggered by *non-rare values* should be detected by traditional ATPG testing. First, a logic simulator is used to find low controllability signals, which results in a set of targets to attach a HT trigger. Second, a fault simulator is used to identify low observability signals, which results in a set of targets for payload. From these two sets, the trigger values and frequencies of each possible HT are computed, as well as the input trigger vectors associated with these trigger values. Then, an ATPG tool is used to check whether each vector from the set can be propagated to the circuit output. Simulations show that, assuming a 2-input HT, ATPG vectors are not sufficient for trigger coverage.

In [9], Chakraborty et al. propose to generate a set of test patterns that maximizes the chance to detect a HT. The proposed methodology is Called MERO (for Multiple Excitation of Rare Occurrence). The assumption is that the number of times a HT trigger condition is satisfied increases with the number of times the trigger signals have been individually excited to their rare value. This results then in increasing the probability to trigger the HT. From the circuit description, a set of random patterns, a list of rare signals and the number of times to activate each signal to its rare value, the set of patterns is modified so that each signal satisfies its rare value the desired number of times. To validate the method, a comparison with random and ATPG patterns has been done. Several HTs have been considered, with 2 or 4 triggers and 1 payload. The conclusions are that the MERO set produces a better HT coverage with a reduced test set.

Salmani et al. in [10] propose to modify the circuit description in order to increase the probabilities of signal when it is lower than a specific threshold. This is done with the insertion of so-called *dummy* scan *flip-flop*s. Since the dummy flip-flops should not have any impact on the functionality of the circuit, they are supplied be either '0' or '1' to avoid changing the functionality of each signal. In scan mode, the output of each flip-flop is supplied by the scan input pin.

III. RARE VALUE BASED LOGIC ENCRYPTION

We propose a hardware combinational encryption technique as those presented in [1, 2], where an external key is added to the circuit so that the circuit operates correctly only if the correct key value is provided. Compared to previous logic encryption approaches, the goal of our encryption is to protect the ICs from mask theft and illegal overproduction as well as help thwarting HT insertion.

The idea is to minimize *rare values* i.e. low controllability signals, such as introduced in [10]. This idea is based on the assumption that it is likely that an attacker will attach a HT on signals having a low controllability. The goal is therefore to prevent the attacker from exploiting the true low controllability signals of a circuit.

This is done by virtually changing the probabilities of each signal to be '0' or '1'. Furthermore, as we do not want the encryption to degrade the performance of the circuit, we make sure that it does not generate a delay overhead. As opposed to the work in [10], in which the modification of the probabilities of the signals was effective only in scan mode, our encryption scheme modifies the probabilities of the signals in functional mode also.

In the following sub-sections we first introduce the method to calculate the probability of a signal to have a particular value, the method we used to calculate the slack time, and finally the proposed encryption algorithm.

A. *Signals with a low controllability*

The first step is to find low controllability signals in a circuit. Even if large simulation results can provide good results, they require extremely high execution time and, moreover, they can lead to imprecise results if only a small portion of input vectors is considered.

In order to be independent of test vectors, an alternative is to compute the probabilities of being '0' or '1' of all signals of a circuit, as presented in [10].

Another alternative, presented in [11], is to use of a boolean functional analysis to identify so-called *nearly-unused* logic. The idea is to determine the influence signals have over outputs of a circuit and then identify the signals that have an abnormally low degree of influence. In that case the focus is made rather on low observable signals than on low controllable signals.

We chose to compute the probabilities of all signals of the circuit as described in [12]. The algorithm consists in propagating the probabilities of each signal to be '0' or '1' from the inputs to the outputs, probabilities of ½ and ½ being put to the inputs of the circuits (cf. Figure 2). To be accurate, this probability-based method takes into account re-convergent signals (i.e. correlations among input signals of a gate) by exploiting the *support-set* of each signal (i.e., the set of signals representing a portion of the circuit that has only independent signals as inputs) as presented in [12].

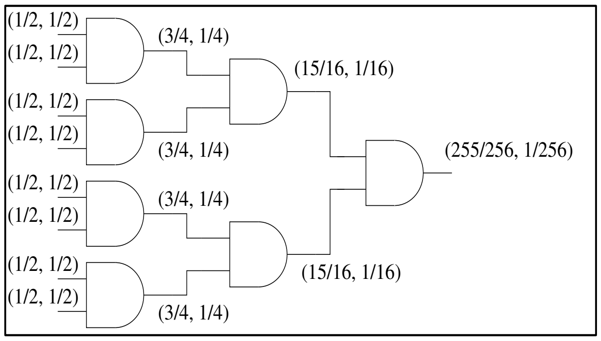

Figure 2. Signals probability.

B. *Slack time*

The proposed encryption method is not intended to degrade the timing of the target circuit. Therefore, the additional gates used for the encryption will be inserted in paths having positive slack time (timing margin). Based on the gates and interconnect delay model, a timing analysis is done to compute the circuit signals slack time. The available slack time analysis is performed as follows (see Figure 3):

- 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),
- Then, the slack time at any timing point is the difference of its required arrival time minus its arrival time.

Once this information is known for each signal, only signals with a slack time large enough to accommodate the insertion of a gate are candidates for encryption.

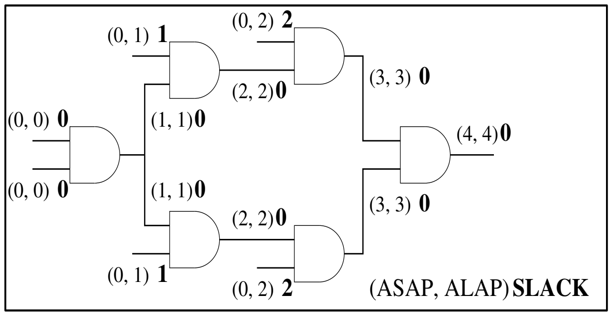

Figure 3. Slack time computation.

C. *Encryption algorithm*

Our encryption algorithm is achieved by inserting AND/OR gates in the design, such as described in Algorithm 1. From a number of gates to include (representing a compromise between quality of encryption and the corresponding overhead of number of gates added), a chosen unbalanced probability (e.g. 0.1%), and a minimum slack (the slack that is necessary to incorporate a gate without changing the overall timing), the algorithm minimizes the number of signals having unbalanced probabilities below the threshold. The algorithm follows these steps:

- Probabilities of having '0'/'1' are calculated for each signal;
- Candidate signals for encryption are the signals preceding a signal with an unbalanced probability (i.e. all the signals connecting to the inputs of the gate driving the signal with the unbalanced probability);
    - Slack times are calculated for each candidate;
    - Remaining candidates are those with a positive slack time;
    - Among the remaining predecessors, the choice is made on the signal with the most unbalanced probability (it is assumed that it is a change in the probability of this signal that will have the greatest impact);
    - The gate to include is chosen depending on the probability that has to be balanced
        - If the probability is close to 0, an OR gate is included and the corresponding key is 0
        - If the probability is close to 1, an AND gate is included and the corresponding key is 1.

Figure 4 shows the structure of the addition of an OR gate aiming at removing a "rare 1" to the following signal. The probabilities of '1' are put along the figure showing that the probability of the output net is multiplied by 10 after encryption.

This encryption procedure produces an identical key for every IC. However, in order to thwart overproduction, the key must be unique to each IC. This aspect is beyond the scope of this paper, interest readers can refer to [1] and [2] in which solutions are proposed. For example, the insertion of a PUF into each IC can be used such as presented in Figure 5.

Algorithm 1 Encryption

```
1: Set NB_GATE the desired threshold
2: Set PROBA_MIN the desired threshold
3: Set SLACK_MIN the desired threshold
4: Compute signals probabilities of circuit c
5: for all signals s of c do
6:     if probability of s < PROBA_MIN then
7:         Append s to list rare_signals
8:     end if
9: end for
10: nb = 0
11: for all signals rs in rare_signals do
12:     if nb >= NB_GATE then
13:         break
14:     end if
15:     for all signals ps preceding rs do
16:         if slack of ps >= SLACK_MIN then
17:             Append ps to list previous_signals
18:         end if
19:     end for
20: Denote signal to encode es the one with the most
    unbalanced probability among signals in list
    previous_signals
21:     if probability of es near 0 then
22:         encoding gate is OR
23:         key is 0
24:     else
25:         encoding gate is AND
26:         key is 1
27:     end if
28:     nb += 1
29: end for
```

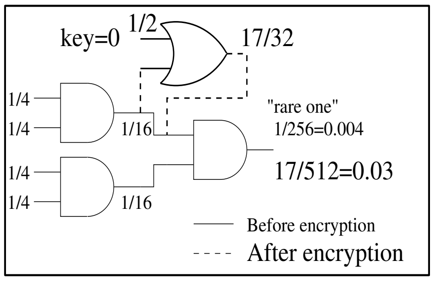

Figure 4. Encryption example

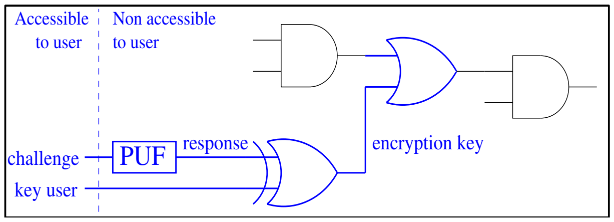

Figure 5. Use of PUF in [2]

IV. Experimental Results

We evaluated our method on a 128 bits AES cipher. We first evaluated the probabilities for all signals of the circuit. Then, we encrypted the circuit, and made a second evaluation. Encryptions were made with an extra cost of 32, 64, 128, 256, 512 and 1024 encryption gates.

A. Probability distribution

Table 1 shows the distribution of the probabilities to be '0' of all signals before encryption. They show the number of signals having a probability to be '0' between 0 and 0.1, 0.1 and 0.2, etc… From the table, 9077 signals have an unbalanced probability below 0.1 or higher than 0.9 (i.e., 68.6% of the signals).

TABLE I. Probabilities of the AES

Probability to be 0	Number of signals
0 -> 0.1	4508
0.1 -> 0.2	533
0.2 -> 0.3	280
0.3 -> 0.4	88
0.4 -> 0.5	321
0.5 -> 0.6	2064
0.6 -> 0.7	61
0.7 -> 0.8	268
0.8 -> 0.9	533
0.9 -> 1	4569

Table 2 focuses specifically on the most unbalanced probabilities of the circuit (below 0.01 and higher than 0.99). The signals with these probabilities are the preferred targets of an attacker. This table shows that 1432 (accounting for 10.8% of the whole circuit signals) signals have a probability of being '0' of 0.004 or 0.996, which are the most unbalanced probabilities of this circuit.

TABLE II. Unbalanced probabilities of the AES

Probability to be 0	Number of signals
0.004	698
0.008	1007
0.992	1107
0.996	734

B. Probabilities after encryption

Several circuit encryptions were made with an extra cost of 32, 64, 128, 256, 512 and 1024 encryption gates. This represents an area overhead of respectively 0.22%, 0.45%, 0.9%, 1.8%, 3.6% and 7.3%.

Table 3 presents the most unbalanced probabilities of the circuits for each encryption. The results show that the goal of our method is achieved: most of the probabilities become less unbalanced. In some cases, an undesirable effect appears: unbalanced probabilities become even more unbalanced. Therefore these signals could be the target for HT insertion. Anyway, these signals behave as expected when the correct circuit secret key is set and a possible HT placed in those signals would become inactive for the correct key.

Moreover, the modification of these unbalanced probabilities have an impact on other signals. This is presented in Table 4, which shows the number of signals for which their probability is modified because of the encryption process. Among the 13225 signals of the original circuit, 717 are affected for a 32 gates encryption and up to 5015 for a 1024 gates encryption. These additional modifications are not the main targets of our encryption, but they represent an interest: they prevent an attacker from exploiting the true probabilities in a circuit to incorporate a HT.

Figure 6 shows, for the 1024 gates encryption, the values of the probabilities of the signals whose probability is modified by the encryption. In this Figure are presented the initial probabilities in dark gray and the corresponding changed probabilities in light gray, for the 5015 concerned signals. Most of the probabilities are improved (i.e., from values close to 0 and to 1, they are shifted to 0.5). Moreover, several probabilities that were not directly addressed by the encryption algorithm are modified, thus making non-exploitable the probability information to insert HTs.

C. Comparison to previous work

Our encryption technique is a combinational encryption as those presented in [1, 2].

In terms of area overhead, the three techniques are similar, except that we incorporate AND/OR gates while XOR gates are incorporated in [1] and XOR/XNOR gates in [2]. The principle is the same in different approaches: area overhead generated by the encryption is a compromise between quality of encryption and number of gates that can be inserted.

In terms of delay overhead, previous approaches do not prevent the encryption to impact critical paths, as opposed to our approach.

Concerning quality of encryption, no relevant comparison can be made since the goals of the encryptions are different.

Our work can also be compared to the one in [10] as both works intend to modify the probabilities of the internal signals of a circuit.

In [10] flip-flops are inserted to increase the *transition probability* of the signals (described as *P0xP1*, *P0* being the probability to be '0' and *P1* to be '1'). Once again, the area overhead depends directly on the quality of encryption (from 0.2% to 5.2% in the presented results).

Besides, the work in [10] avoids inserting the flip-flops on critical paths not to affect the delay of the circuit whereas our approach is limited to signals with a positive slack.

In [10], the encryption ensures that all signals in a circuit have their transition probability greater than or equal to a given threshold. The bigger the threshold is, the greater the number of signals to encrypt is. This is an interesting variant of our approach that should be studied in future work. Furthermore, this would make possible to perform a more quantitative comparison of these two approaches.

V. Conclusion

Our proposed scheme is the first approach to use logic encryption in order to help preventing unauthorized overproduction as well as help thwarting HT insertion.

The goal of our encryption technique is to minimize the number of *rare values* in a circuit. That way, an attacker cannot exploit these rare values to insert a HT. The goal is to make the insertion more difficult, if not impossible. Either the attacker determines that controllability is not low enough and avoids putting a HT trigger. Either he considers the controllability low enough and still puts the trigger. In this latter case, we assume that the encryption would help to activate the HT during test. This scenario remains to be studied in future work.

From several encryptions from 32 to 1024 gates, simulation results show the many changes in signals probabilities in an AES cipher. Not only unbalanced probabilities are minimized, but also balanced probabilities are modified. This shows the quality of the encryption, which is essential to prevent an attacker from inserting a stealthy HT.

Acknowledgments

This project has been funded by the French Government under grant FUI-BPI-France F1211008Z **HOMERE** (**H**ardware tr**O**jans : **M**enaces et robust**E**sse des ci**R**cuits int**E**grés).

TABLE III. UN BALANCED PROBABILITIES OF THE AES AFTER ENCRYPTION

Probability to be 0	Number of signals based on each encryption						
	Initial	32	64	128	256	512	1024
0.002	0	0	0	0	0	0	2
0.004	698	683	664	641	573	436	187
0.006	0	0	1	5	13	35	67
0.008	1007	1003	998	979	957	915	845
"Rare 0"	1705	1686	1663	1625	1543	1386	1101
0.992	1107	1102	1094	1086	1062	1011	931
0.994	0	0	0	1	2	4	10
0.996	734	718	702	660	597	473	202
0.998	0	0	2	5	8	21	50
"Rare 1"	1841	1820	1796	1749	1669	1509	1193

TABLE IV. MODIFIED PROBABILITIES OF THE AES

Number of signals based on each encryption					
32	64	128	256	512	1024
717	1145	1653	2419	3646	5015

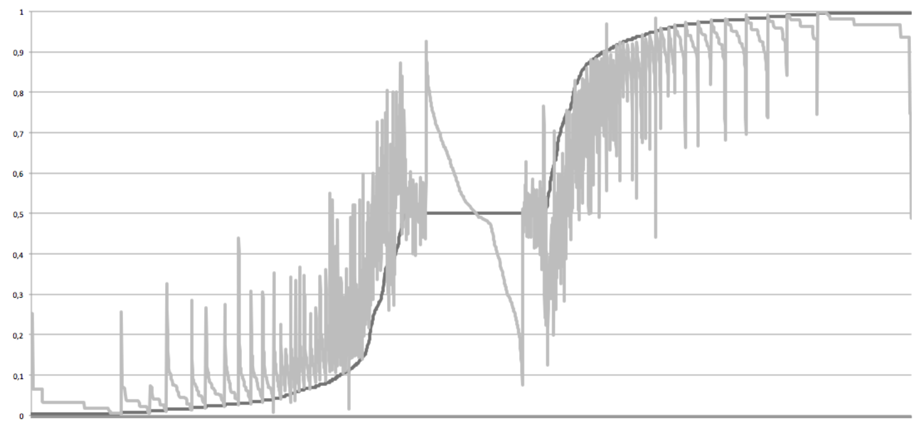

Figure 6. 5015 probability changes of the 1024 gates encryption (before encryption in dark grey, after encryption in light grey).

REFERENCES

[1] J. A. Roy, F. Koushanfar and I. L. Markov. EPIC: Ending Piracy of Integrated Circuits. In *Design, Automation and Test in Europe (DATE'08)*, pages 1069–1074, 2008.

[2] J. Rajendran, Y. Pino, O. Sinanoglu and R. Karri. Logic Encryption: A fault Analysis Perspective. In *Design, Automation and Test in Europe (DATE'12)*, pages 953–958, 2012.

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

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

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

[6] 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)*, pages 1362–1365, 2008.

[7] 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),* pages 296–310, 2007.

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

[9] 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),* pages 396–410, 2009.

[10] 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.

[11] A. Waksman, M. Suozzo, and S. Sethumadhavan. FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis. *ACM SIGSAC Conference on Computer & Communications Security (CCS'13)*, pages 697–708, 2013.

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