Hardware Trust through Layout Filling: a Hardware Trojan Prevention Technique

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

Hardware Trust through Layout Filling: a Hardware Trojan Prevention Technique

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

***Abstract*— The insertion of malicious alterations to a circuit, referred to as Hardware Trojans, is a threat considered more and more seriously during the last years. Numerous methods have been proposed in the literature to detect the presence of such alterations. More recently, *Design-for-Hardware-Trust* (DfHT) methods have been proposed, that enhance the design of the circuit in order to incorporate features that can either prevent the insertion of a HT or that can help detection methods. This paper focuses on a HT prevention technique that aims at creating a layout without filler cells, which are assumed to provide a great opportunity for HT insertion, in order to make the insertion of a HT in a layout as difficult as possible.**

Keywords- Harware-Trojans; Design-for-Hardware-Trust; Layout; Placement and routing

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 the Integrated Circuits (ICs) industry. This raises the question about untrusted foundries in which the insertion of malicious circuitry, referred to as Hardware Trojans (HTs), is a possible threat [1, 2]. A wide variety of HTs can indeed be implemented for altering the initial functionality of a design. They differ in their phase of insertion in the flow, physical characteristics, activation mechanisms, or functionality [3]. The challenge lies in how to detect HTs and/or prevent HT insertion knowing the stealthy nature of that threat. Few hundred transistors are indeed sufficient to insert a malicious behavior in a billion-transistor design. There is therefore a need of developing novel techniques to secure the ICs against this threat [4].

Many studies have been recently dedicated to the HT threat. Proposed techniques can be classified into two main categories: detection methods and DfHT. In the former case, the design flow of the circuit is not modified and the circuit is tested after its fabrication in order to ensure the nonexistence of HT. On the contrary, DfHT methods consist in enhancing the design of the circuit in order to incorporate features that can either prevent the insertion of a HT or that can help detection methods [5].

HTs detection methods are divided into two types: *side-channel analysis* [6, 7], and *logic testing* [8-11]. *Side channel analysis* methods focus on monitoring physical parameters of the circuit, such as the power consumption [6] or path delay [7]. Relying on golden ICs (i.e. circuit that have been ensured to be HT-free by destructive methods), a comparison is made with the circuits under test. The assumption is that the introduction of any additional malicious logic would increase power consumption or some path delay. The main weakness of side-channel analysis is to manage process variations. This makes them hardly effective on small HTs. Most *logic testing* based methods focus on so-called rare values based HTs i.e. HTs that are dormant until a very rare condition activates/triggers them [8]. The HT payload is then observed with an error on the outputs. The main concern is then to be able to activate potential HTs at test time i.e. find test vectors that can maximize the chances of activating HTs [9, 10]. More recently, it is assumed in [11] that an attacker may not have control on the internal signals of a circuit and that he will rather attach a HT trigger on the inputs. The goal is then to produce a reduced set of test vectors using combinatorial testing. Logic testing methods reach their limits when the needed set of test vectors increase to an unaffordable size. They can therefore be hardly effective when considering large HTs requiring the control of numerous signals.

Given the limitations of HT detection methods, the idea of modifying the design flow has emerged. These DfHT methods incorporate into circuits features that can help detection methods or/and make more difficult the insertion of a HT [12-20].

In this paper, we focus on a DfHT method that aims at preventing an attacker in an untrusted foundry from inserting a HT at the layout level. The proposed method consists in creating a circuit layout as dense as possible. As shown in [12], a possible threat for easy HT insertion is indeed provided by the filler cells. Filler cells are inserted in the empty spaces of the circuit layout after the placement step and do not have any specific logic function. Their goal is to improve the density uniformity of the circuit [21]. They can provide a great opportunity for HT insertion because they are not tested after production. Therefore they can be

978-1-4673-9039-2/16 $31.00 © 2016 IEEE
DOI 10.1109/ISVLSI.2016.22

easily removed and replaced by a HT. Furthermore, removing filler cells to replace them with new gates seems stealthier from a visual inspection point of view [22] than modifying the original functional cells of the circuit. In this way, any intentional modification of the placement of the logic gates becomes extremely hard to be performed by a possible attacker.

The authors of [12] proposed a technique called Built-In Self-Authentication (BISA) in which interconnected combinational cells are used as filler cells in order to create an additional combinational network. By testing the extra network (besides the original design), it is possible to understand if filler cells have been altered. The test of the additional network is performed through a Built-In Self-Test architecture, where Linear Feed Back Shift Registers (LFSRs) and Multiple-Input Shift-Registers (MISRs) are also implemented in the space dedicated to filler cells. Based on the same insight, it is proposed in [13] to improve the method by prioritizing the empty spaces to fill, since achieving 100% occupation ratio is in most cases impossible for routability reasons. The proposed idea is to fill in priority the so called "critical empty spaces" i.e. the ones that are close to signals which are prone to be selected for HT triggering. Furthermore, it is also proposed in [13] to use shift registers in order to test the added combinational functions, instead of a TPG and a MISR, with the intention of needing less space left. However, this method still reaches its limits because of the large size of the FFs. Firstly, there may be a lack of FFs inserted with regards to the number of combinational cells inserted. Secondly, when the original occupancy rate reaches about 85% in medium size ASICs, no space is large enough to hold a FF.

The contribution of this paper is to enhance the method proposed in [13], by:

1. Providing an enhanced algorithm that allows deriving the best number of combinatorial functions given the possible number of FFs to insert;
2. Describing experiments consisting in inserting only logical gates when a design is too dense to contain any FF;
3. Describing resulting circuits in terms of power consumption and timing.

The rest of the paper is organized as follows: Section II presents related works on HT prevention. Section III details the proposed layout level design approach. Experimental results are presented in Section IV. Finally, Section V concludes the paper.

II. Related Works

A. RTL level

Chakraborty et al. propose in [14] to modify Finite-State Machines (FSMs) in order to create a special mode of system operation called "transparent mode" that allows to control low-controllability signals and observe low-observability signals. A key port is added to the circuits, and on the application of the right sequence of keys, the FSM enters the transparent mode.

The method proposed in [15] consists also in adding a key, but in that case, the goal is to obfuscate the FSM: upon activation, the circuit is in an "obfuscated mode", and enters the normal mode only upon application of the right input sequence of keys.

B. Gate level

The approach proposed in [16] inserts so-called "dummy flips-flops" in order to improve the controllability of the design and thus remove rare triggering condition for HTs.

The method in [17] aims also at removing rare triggering conditions thanks to the insertion of AND/OR gates controlled by a key. These gates have the double feature of changing the controllability of the signals (in order to remove low controllable signals) and obfuscating the functionality of the circuit: the circuit behaves correctly only upon the application of the right key.

C. Layout level

In [18], controllability is improved thanks to an inverted voltage scheme. Voltage inversion on a CMOS gate indeed changes the gate behavior e.g. a NAND gate behaves like a AND gate, a NOR gate behaves like an OR gate. Consequently, while the NAND gate output is initially low controllable to '0' and easily controllable to '1', the voltage inversion on that gate makes its output easily controllable to '0' (and low controllable to '1'). Using both voltage configurations allows controlling the gate output to both values. This approach affects the place and root process since gates with the same combinatorial depth must be connected to the same voltage supply network, and gates in the alternate levels (combinatorial depths i and i+1) must be connected to separate supply voltage network.

D. Transistor level

Methods in [19, 20] propose to change the standard-cell library used.

It is proposed in [19] to implement circuits with differential cascade voltage switch logic (DCVSL), which produces complementary logic values in all signals. The assumption is that the insertion of a HT will necessarily lead to non-complementary inputs in a DCVSL gate and consequently abnormal short-circuit power peaks. Note that any HT implementation leading to produce errors simultaneously on complementary values will not be detected.

The idea of creating new types of CMOS gates that include not only the wanted functionality but also a so-called dual functionality is introduced in [20]. The aim of this dual functionality is that a slight change caused by a HT in the primary functionality should cause a large difference in the dual functionality. A potential HT should therefore be easily detected by logic testing methods. This hypothesis, interesting in theory, has unfortunately not been implemented.

E. Synthesis

As mentioned before, Design-for-Trust methods are useful, either to help detection methods or to make HT insertion more difficult. Methods at RTL or gate levels provide effective protection, but need to incorporate new functionality into the circuits and therefore entail silicon area overhead. Method proposed in [19, 20] require creating the ICs with uncommon logic cells, which can be very costly and lead to a non-negligible area/power consumption overhead.

The only methods that do not generate larger circuits are the ones based on "layout filling", as proposed in [12, 13]. Filling unused spaces with functional cells instead of filler cells has indeed no impact on the silicon area. Furthermore, these methods are not costly or time consuming since they are based on a standard library as well as standard place and route tools. Although promising in theory, this idea may nevertheless be limited by the place/route tools capabilities. Placement and routing are indeed critical steps in the VLSI design flow. Limited routing resources (in terms of number of available wire tracks) are indeed the initial cause of the need to enlarge the layout during placement to provide enough wire tracks to resolve routing congestion, hence the creation of the "empty spaces" between the standard cells. Filling these empty spaces with functional cells and connecting these additional cells together may generate new routing constraints, which may lead to unroutable layouts. That is why the idea of prioritizing the empty spaces to fill was introduced in [13]. However, the method proposed in [13] reached its limits because of the size of the shift registers to add. We address this issue in this paper by proposing an enhanced procedure in which an iterative analysis to derive the best way to connect the combinatorial cells together given the possible number of FFs to insert.

III. Proposed Layout-level HT prevention approach

Fig. 1 shows the block diagram of the proposed layout level design approach. The principle of this method is to fill the "empty spaces" needed to perform routing with extra combinatorial functions. Thus an attacker has no room in the design to add extra functions such as HTs.

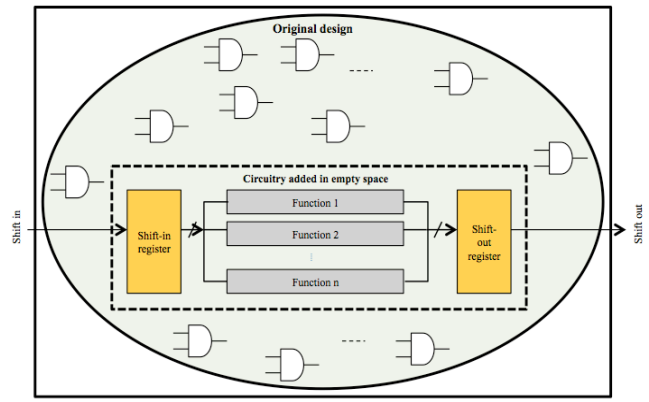

Figure 1. Block diagram of an IC with HT prevention

Extra standard-cells are inserted and connected together to form several combinatorial functions that are independent from the original design. A significant constraint is to create testable functions. The goal of the method is indeed to test these functions on the fabricated ICs to ensure that no function has been modified by an attacker introducing a HT. Although it is difficult for an attacker to identify additional functions from the original circuit, the pessimistic scenario in which the attacker succeeds is therefore handled. In order to test these functions, shift-registers are used at both input and output sides in order to apply input patterns and receive output responses. These shift-registers are also implemented within unused empty spaces.

The next sub-sections detail the global flow, as well as the several steps of the method: the identification of the so-called "critical empty spaces" in the layout, the filling with standard cells and shift registers and the building of the extra functions.

A. Global flow

From a placed circuit (and a chosen initial occupation ratio), the global flow of the experimental procedure is as follows:

1. Computation of the empty spaces and critical empty spaces,
2. Placement of the maximum number of FFs that can fit into these empty spaces (possibly none) to create the shift registers,
3. Placement of the logic cells in the remaining empty spaces (giving priority to critical empty spaces),
4. Interconnections of cell in order to create the appropriate number of logical functions depending on the number of FFs (only 1 function in case of no FF inserted).

Then the whole circuit is routed (original circuit and added functions).

The aim of this procedure is to find the maximum occupation ratio possible that allows routing. The procedure described is then used iteratively, with a goal ratio of 100%, which is decremented until no routing violation occurs.

Once the ICs are fabricated, a test phase is required before they are deployed in the field. In addition to the conventional tests ensuring the proper operation of each IC, it is necessary to test the additional functions, to ensure that a HT was not inserted. To do so, input patterns are shifted in thanks to the input shift register, and the responses of all combinatorial functions are shifted out.

One has to notice that the shift-registers are using a separate clock than the original circuit. Firstly, it allows to switch of the additional functions once in the field in order to prevent unnecessary power consumption. Secondly, it allows relaxing the timing constraints of the additional functions, which will also help relaxing the constraints for routing these functions.

B. *Identification of critical empty spaces*

As mentioned before, even with a filling method as the one we propose, being able to always achieve 100% occupation ratio is not possible given routing limited resources. We therefore propose to fill in priority so called "critical empty spaces" as follows.

"Critical empty spaces" are the empty spaces that are close to signals with a large slack (i.e. time margin). The reason is that an attacker is likely to insert a HT in that kind of empty spaces in order take advantage of the large slack of these signals to insert a HT's trigger. These signals are indeed insensitive to HT insertions from a delay point of view i.e. the insertion of a HT will not result in any degradation in the timing performance of the original design and will be insensitive to HT detection techniques based on delay measurements [10]. Therefore, these critical empty spaces will be considered as a priority during filling.

C. *Filling*

First of all, flips-flops that will be used to create shift-in and shift-out registers are inserted. The flip-flop being a very large cell, it is introduced first in order to use all the "big empty spaces". As will be seen in the experimental results section, due to the large size of flip-flops, the number of flip-flops introduced is often not sufficient with respect to the number of combinatorial gates. Thus in this step, we insert as many flip-flops as possible.

Then, left critical empty spaces are filled with combinatorial cells, from larger to smaller ones. Last, left empty spaces are filled the same way. The choice of the combinatorial cells to use is done according to each cell's a) width, b) number of inputs, and c) decoupling capacitance value. For an empty space of a given size, the choice is firstly restricted to the larger cells fitting into this empty space (in order to limit the number of introduced cells). Then, cells with a large number of inputs have priority (because it helps reduce the size of the functions as described afterwards). Last, cells with larger decoupling capacitance values have priority because they help to compensate the absence of filler or decoupling capacitance (DECAP) cells [26]. For instance if we consider different cells with different parameters from Table I, the OR4 gate is the most suitable one for an empty space of 5um. Note that, cells with large fan-outs are presented in Table I. One can choose not to use such cells in order to prevent possible resizing attacks, or to use them (under the assumption that an attacker will not be able to perform such an attack).

Besides, if an empty space can contain only an inverter, we choose to let it empty: this kind of empty space will not usable by an attacker. Furthermore, if we made the choice to insert an inverter and use it in the creation of the combinatorial functions, it would generate additional unnecessary contraints on the routing.

D. *Construction of the extra functions*

Once added in the layout, the cells are connected in a tree structure to build up the functions, as shown in Fig. 2. The process iterates, from inputs down to outputs until a function with 1 output is created. In order to prevent routing congestion, the first cell of each function is chosen as close as possible to one flip-flop and other cells used to build the function under construction are selected from the first one's closer cells.

In order to prevent an attacker from replacing a function with a HT, a constraint is to consider: not create two identical functions. This would give an attacker the opportunity to replace one of the two functions (and connect together the two outputs) without this being visible during the test phase.

Besides, as mentioned before, this method can face a lack of FFs inserted relative to the number of logical functions created. Given that the number of functions inputs is directly related to the number of FFs in the shift in register, and the number of functions to the number of FFs in the shift out register, the idea is to find the optimal function size (i.e. inputs number) i.e. the size that produces an optimal number between the number of inputs and outputs, resulting in the smallest number of FF possible.
In practice, we iteratively run the global flow for several input numbers, observe the resulting number of functions and then choose the optimal one.

I. EXPERIMENTAL RESULTS

We evaluated our method on several benchmarks. Experiments were conducted with a 65nm library and Synopsys tools for synthesis, placement and routing.

TABLE I. CELL SELECTION

Function	Width (um)	Input Count	DECAP
NAN3X38	4.6	3	0.67
AND4X25	4.6	4	0.44
AOI12X5	5	3	0.61
AOI21X35	5	3	0.61
OAI12X37	5	3	0.65
OAI21X37	5	3	0.65
OR4XX29	**5**	**4**	**0.51**

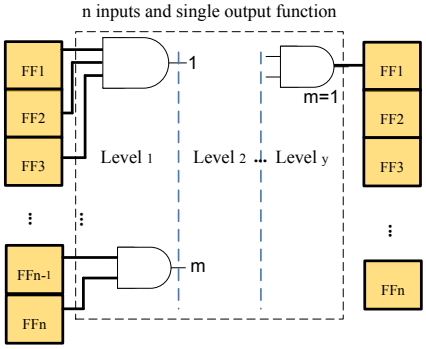

Figure 2. Function formation

TABLE II. LAYOUT FILLING (1)

Benchmark	Initial OR	Initial leakage power (nW)	Initial DAT (ns)	Final OR	Final leakage power (nW)	Final DAT (ns)	FFs/cells	Inputs/ Functions	Left-over cells	NAND possible
AES	75	173.59	5.18	91	191.56 (+10%)	5.22(+0.8%)	61/1036	30/31	27	280
	80	173.95	5.11	90	185.13 (%6%)	5.25 (+2.8%)	27/609	12/14	31	237
	85	173.59	5.12	88	175.6 (+1%)	5.12 (-0.2%)	6/172	5/1	51	202
S13207	75	14.31	0.89	95	23.51 (+64%)	0.6 (-32%)	29/125	19/10	0	0
	80	13.42	0.91	90	18.41 (+37)	0.58 (-36%)	13/80	5/8	9	55
	85	13.42	0.92	90	14.99 (+12)	0.92 (+0%)	7/39	2/5	6	51
S35932	75	118.14	1.19	95	175.24 (+48%)	0.77 (-35%)	277/977	62/26	0	139
	80	117.91	1.17	91	160.8 (+36%)	0.71 (-39%)	131/504	11/70	0	383
	85	117.15	1.13	90	145.69 (+24%)	0.73 (-45%)	112/217	10/34	0	552
RSA	75	35.99	9.8	93	45.35 (+26)	9.84 (+0.3%)	62/328	17/23	0	45
	80	35.92	9.82	93	43.26 (+20%)	9.63 (-2%)	53/209	46/7	0	39
	85	35.86	9.72	92	38.86 (+8%)	9.9 (+1.8%)	16/114	10/6	1	89
RS232	75	18.28	1.9	91	22.63 (+24%)	1.84 (-4%)	32/157	27/5	0	25
	80	18.29	1.9	91	20.95 (+15%)	1.83 (-4.7%)	19/102	15/4	0	28
	85	18.26	2.1	87	18.72 (+2)	1.92 (-8.7%)	4/17	3/1	12	60
ARM4U	75	41.49	9.71	93	50.26 (+21%)	10.79 (+11%)	70/328	63/6	0	42
	80	41.48	9.86	91	46.98 (+13%)	10.77 (+9.3%)	36/191	16/20	0	79
	85	41.45	9.8	91	43.6 (+5%)	10.52 (+7.4%)	22/97	13/8	0	81

Table II presents the results of layout filling in terms of: (1) initial occupation ratio (OR) and final OR reached (i.e. the densest OR without routing violation), (2) initial and final leakage power (note that since the extra testable functions are connected to a separate clock, they are off during normal operation and therefore do not increase dynamic power) and (3) initial and final data time arrival. To comprehend the filling of the circuits, the number of FFs and logical cells inserted is presented as well as the number of functions created (along with the number of inputs for each function) and the number of combinatorial cells that remain unconnected due to a lack of FFs. Last, to better evaluate the difficulty for an attacker after applying our method, the number of NAND gates that could be inserted after filling is presented.

High occupancy rates (above 90%) can be achieved by our method. Besides, the final OR achieved is generally the largest from the minimum initial OR and the bigger the initial OR, the greater the risk of a lack of FFs. Leakage power is degraded as expected, in proportion to the number of added cells: the bigger the initial occupation ratio, the less the deterioration. Data arrival time is degraded or enhanced. It totally depends on the routing algorithm; no prediction can be made. The number of exploitable spaces remaining after filling may seem too large in some cases, however, this number is in most cases far smaller than the number of cells inserted. This shows that the method has removed a majority of opportunities an attacker

A layout of the benchmark s35932 cipher is presented in Fig. 3 in which the circuit is placed with an OR of 75%. The cells in yellow/bold are the cells added by our method with a goal OR of 100%.

Table III presents results in case of an initial OR too large to insert any FF. To test the limits of the method, the initial OR chosen is the maximum OR possible without routing violation. Our method can cope with such constraints, since it manages to insert logical cells and to create a logical function in all cases, even achieving 100% in 2 cases.

Fig. 4 shows the interest of seeking for the best combination between the number of combinatorial functions and the number of inputs for each function i.e. the number of inputs that leads the least amount of non-connected logical cells, depending on the possible number of FFs. In this figure, the number of remaining unconnected cells is presented, according to the number of inputs chosen for the combinatorial functions. These data correspond to the filling of the ARM4U benchmark, from 80% to 91%. 36 FFs can be inserted. All numbers of functions inputs have been tested from 4 to 35, and only one solution allows to have no cells remaining not connected: 16 inputs. Thanks to this method, our method can be applied on circuits with an initial rate up to 85% (only 80% were reached in [15]).

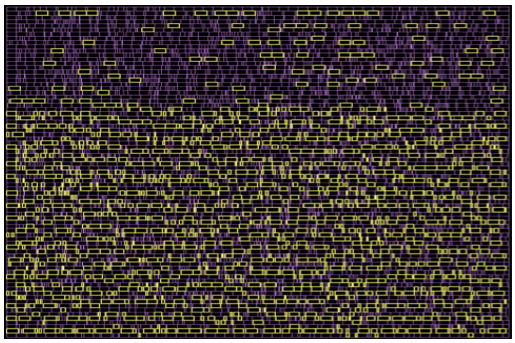

Figure 3. Layout occupation (added cells in yellow/bold)

TABLE III. LAYOUT FILLING (2)

Benchmark	Initial OR	Final OR	Number of Inputs
AES	93%	94%	19
S13207	99%	100%	8
S35932	99%	100%	30
RSA	94%	95%	9
RS232	95%	96%	9
ARM4U	96%	97%	10

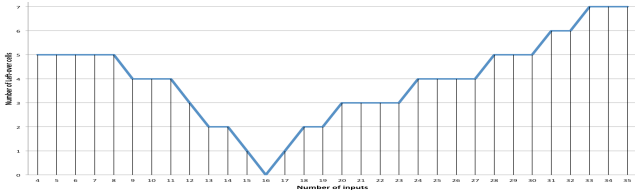

Figure 4. Iterative analysis of the functions size

II. CONCLUSION

In this paper, we have presented a DfHT method that aims at creating a layout as dense as possible in order to prevent possible HT insertion at layout level by an attacker in an untrusted foundry. The method consists in filling empty spaces in a layout by functional cells instead of filler cells. These additional functions are testable in order to prevent an attacker from replacing them with a HT. Such a method can generate large constraints for the routing; we explained how to minimize these additional constraints. Experimental results show that very high occupancy rates can be achieved, which demonstrates the feasibility of the method. To avoid potential degradation of the critical path due to routing, a future work could be to develop an ad-hoc routing algorithm, in order to route the additional functions after the routing of the initial circuit, i.e. without modifying the initial routing.

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] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor and Y. Makris, "Counterfeit integrated circuits: a rising threat in the global semicondictor supply chain", In Proceedings of the IEEE, Special Issue on Trustworthy Hardware, 102(8):1207–1228, 2014.

[3] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection", IEEE Design & Test of Computer, 27:10– 25, 2010.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[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), pp. 396–410, 2009.

[10] 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.

[11] P. Kitsos, D. E. Simos, J. Torres-Jimenez and A. G. Voyiatzis, "Exciting FPGA cryptographic trojans using combinatorial testing", In IEEE International Symposiul on Software Reliability Engineering (ISSRE'15), 2015.

[12] K. Xiao and 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.

[13] P.-S. Ba, P. Manikandan, S. Dupuis, M.-L. Flottes, G. Di Natale and B. Rouzeyre, "Hardware trojan prevention using layout-level design approach", In IEEE European Conference on Circuit Theory and Design (ECCTD'15), 2015.

[14] 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.

[15] R. S. Chakraborty and S. Bhunia, "Security against hardware trojan attacks using key-based design obfuscation", In Journal of Electronic Testing, 27(6):767–785, 2011.

[16] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time", In IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 20(1):112–125, 2012.

[17] 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 IEEE International On-Line Testing Symposium (IOLTS'14), 2014.

[18] M. Banga and M. S. Hsiao, "VITAMIN: voltage inversion technique to ascertain malicious insertions in ICs", In IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'09), pp. 104–107, 2009.

[19] W. Danesh, J. Dofe and Q. Yu, "Efficient hardware trojan detection with differential cascade voltage switch logic", In VLSI Desing, Special Issue on Advanced VLSI Architecture Design for Emerging Digital Systems, 2014.

[20] Y. Alkabani, "Trojan immune circuits using duality", In Euromocro Conference on Digital System Design (DSD'12), pp. 177–184, 2012.

[21] J. Ichimiya, "Layout design method of semiconductor integrated circuit, and semiconductor integrated circuit, with high integration level of multiple level metalization", US Patent 7,076,756, 2006.

[22] S. Bhasin, J.-L. Danger, X. T. Ngo and S. Guilley, "Hardware trojan horses in cryptographic IP cores", In Fault Diagnostic and Tolerance in Cryptography (FDTC'13), pp. 15–29, 2013.