2009 22nd International Conference on VLSI Design

A Novel Sustained Vector Technique for the Detection of Hardware Trojans

Mainak Banga and Michael S. Hsiao
Bradley Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, Virginia - 24061
Email: {banga, mhsiao}@vt.edu

***Abstract*—Intentional tampering in the internal circuit structure by implanting Trojans can result in disastrous operational consequences. While a faulty manufacturing leads to a nonfunctional device, effect of an external implant can be far more detrimental. Therefore, effective detection and diagnosis of such maligned ICs in the post silicon testing phase is imperative, if the parts are intended to be used in mission critical applications. We propose a novel sustained vector methodology that proves to be very effective in detecting the presence of a Trojan in an IC. Each vector is repeated multiple times at the input of both the genuine and the Trojan circuits that ensures the reduction of extraneous toggles within the genuine circuit. Regions showing wide variations in the power behavior are analyzed to isolate the infected gate(s). Experimental results on ISCAS benchmark circuits show that this approach can magnify the behavioral difference between a genuine and infected IC up to thirty times as compared to the previous approaches.** [1]

I. Introduction

With the decreasing per component cost for silicon ICs, companies are searching for new avenues to reduce the manufacturing cost. This has led to the outsourcing of the fabrication process. Consequently, the question of security and integrity of the embedded product comes forth as a prime concern. Thus, the design company has to ensure that no subtle intentional alterations had been subjected to the original logic during fabrication. The tiny circuits that are implanted to the original design to make it work contrary to the expected in certain rare and critical situations are called as *Trojans*.

Trojans have been common in the software domain and are commonly referred to as *virus*. Although *viruses* and *Trojans* are not exactly the same, their end consequences are similar. *Viruses* are necessarily malicious and interfere with the normal operation of the host on which they reside, whereas *Trojans* are passive monitors for most part of their operational life cycle until they are triggered. Solutions to counteract *virus* attacks exist in the form of *anti-virus* softwares. But currently there are no such remedies for Trojan attacks in hardware.

Trojans have distinguishing features that make them unique. They are stealthy in nature, which implies that they do not manifest their presence in normal operational conditions of the IC. This also suggests that they are not associated with internal gates that are either highly controllable or highly observable. They are very small in size, occupying only a small fraction of chip area which enables the third party vendor to accommodate them in the same die without altering the physical dimensions of the chip. Although test patters generated from an ATPG can detect most of the manufacturing faults, no such scheme is available to uncover the Trojans because they have an unknown triggering scenario which is difficult to assess and occurs rarely. Additionally, Trojans can have varying spatial locations on the IC and different logical behaviors (counter-based Trojans, sequence-detector Trojans etc.) [6] which complicate the detection mechanism. Finally, Trojans may not affect any of the primary outputs even if they are triggered. As a result, irrespective of whether a Trojan resides in an IC or not, there may not be a difference in the circuit's output behavior. Trojans can be selectively implanted and its absence in one IC does not guarantee its absence on any other. So destructive testing is also not a viable option. On one hand, destructive testing incurs a yield loss where a chip that has been cut open for analysis has to be discarded, while on the other it cannot guarantee genuineness of the other parts not subjected to such testing.

Moreover, on-chip testing structures like *Built In Self Test* (BIST) are also common to check on chip defects and reduce the test time [7], [8], [9]. However, similar to the problems with scan-based testing, BIST patterns may not be able to trigger the embedded Trojan which requires a specific sequence of input data. Hardware security based on *cryptography* and *public and private key* has been prevalent in industry. *Physically Unclonable Functions* (PUF) based structures have been proposed recently [11], [12] to characterize individual IC security key. While PUF based schemes are very effective in preventing external attacks to extract out the internal information from the ICs, Trojan attacks are on-chip intrusions and hence require a different approach. The intelligent nature of Trojans make them immune to such conventional checking procedures. This has directed the researchers to search for newer methods to detect the presence of Trojans. Since one has a limited access to the logical behavior of the device (only at the inputs and the outputs), researchers have identified the use of physical characteristics such as power, circuit delay or radiation behavior to act as a signature for the IC. Methods based on side channel signal analysis have been used in [1], [2] where the authors have used a random sequence of test patterns to differentiate between the actual and the Trojan circuits. However, the magnitude in the difference between the circuit under test (CUT) could be very small and may not be detectable considering process variations. With the process geometries sinking down in the nanometer regions, leakage and process variations continue to increase. Thus it is of utmost priority that the discrepancies in the genuine and maligned ICs

[1]Supported in part by NSF grant 0840936

are highlighted as much as possible so that their probability of detection increases.

In [3], [4], the authors have proposed partition mechanisms to isolate parts of the circuitry that might account for the Trojan. Results show that these methods are helpful in filtering out the infected regions. However, in those papers the Trojans are assumed to be associated with flip-flops of the circuit. This is a restricted assumption since embedded Trojans can monitor any signals in the circuit. Moreover, in [4], the regions under consideration are filtered out based on the flip-counts. A Trojan based on gate inputs may be missed using these techniques.

In this work, we propose a sustained vector methodology that magnifies the power consumption differences between the actual and the Trojan circuitry to a value which is much higher than the process variation. In certain cases the power differential can vary by more than an order of magnitude. Each vector is repeated multiple times to both the genuine and the Trojan-embedded circuits that ensures the reduction of toggles within the genuine circuit. This is needed so that the power dissipation outside of the Trojan will not drown out the extra power from the Trojan. In addition, we propose a scheme to suggest the locations susceptible to Trojan implantations. Regions showing wide variations in the power behavior are analyzed to isolate the infected gate(s). Our method is generalized for detecting Trojans that may be connected to any gate(s) in the circuit. There is no pre-silicon on-chip processing requirement which means that the methodology has no silicon overhead.

The rest of the paper is organized as follows. Section II gives the background on side channel analysis and kinds of Trojans. Section III details our approach. Section IV describes the Trojans inserted in our experimental setup. Section V discusses the results, and Section VI concludes the paper.

II. Preliminaries

In this section we present a brief overview of the concepts and terms that we will use throughout this paper.

A. Side Channel Analysis

With the improvement in cryptographic algorithms it is becoming increasingly difficult for the attackers to infringe the system dynamics in the conventional manner. This has forced them to explore other methods to access internal information. Physical parameters such as timing information, power consumption or electromagnetic leaks (side channel signals) have been shown to provide valuable information about the internal operations going on inside the system. Since side channel signals contain useful information which can reveal the internal functionality of the operating device, it is employed as a powerful non-destructive method of system analysis. The process of extracting internal data from a concealed system by tracking one or more side channel signals is called as *side channel analysis*.

In our work we have used power consumption as our side channel signal. The total dynamic power consumed in an IC is proportional to the operating frequency *f*, the switching capacitance *C* and the supply voltage *V* and are related by the following equation [5]:

$$\mathbf{P} = \mathbf{CV^2f} \tag{1}$$

As the supply voltage *V* and the operating frequency *f* remain constant for an IC, the parameter for analysis is the switching capacitance *C*. This depends on the number of gates that undergoes toggle(s) in a system for any given vector pair.

B. Trojan Types

Combinational Trojan: An embedded combinational circuit that monitors static signal conditions with no memory of the signal's previous conditions.

Sequential Trojan: An embedded finite state machine (FSM) that is triggered upon the appearance of a specific sequence(s) of internal signal conditions.

The Trojans in our experiments are sequential Trojans and have been further categorized according to the toggling frequency of the signals from which they receive their inputs. More details will be given later in the paper.

C. Power Profile

A power profile represents the pattern of power consumption in a system. Power consumption for any pair of vectors is dependent on the total number of gates that switch which accounts for the switching capacitance (other factors in Equation 1 remaining the same). In our paper we have used the terms activity profile or power profile interchangeably because number of gate switches in a circuit is directly proportional to the dynamic power consumed by it. Also, a *gate* refers to any combinational or sequential element in the CUT.

III. Our Approach

There are two steps in our approach. The first step aims to detect the presence of a Trojan while the second tries to isolate the region within the circuit that may contain it. We call the first step as *Toggle Minimization* and the second as *Infected Region Isolation* respectively.

A. Step 1: Toggle Minimization

In the context of earlier discussion, power consumed in a circuit depends on the amount of signal switchings for any given vector pair. Since Trojans are minuscule circuits relative to the entire circuit, it is intuitive that the power consumed by the Trojan will also be very small. To observe the extra power that is contributed by the Trojan circuit over the genuine circuit, it is essential that the overall power consumption in the genuine circuit should be minimized. This would highlight the power contributed from the Trojan circuit which is the key to detect its presence. In the process, we also need to ensure that there is at least some kind of activity going on inside the circuit. In other words, the circuit should not be allowed to enter some *sleep mode*, which may also make the Trojan dormant. Sophisticated tools to measure very low on-chip currents are available which can sample the input power pins of the chip and measure the current [13].

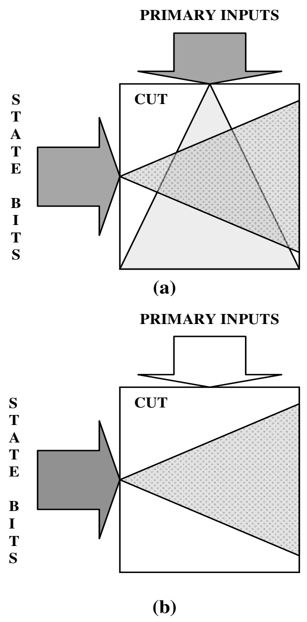

Fig. 1. Concept of activity minimization. In (a) circuit activity is created by both flip-flops and PIs whereas in (b) only by flip-flops

Circuit activity within the combinational frame of the circuit is induced in two ways: (1) with the changing inputs and (2) with the changing state. While primary inputs are fully controllable, the state variables are not. In order to limit the switching activity within the circuit, we can restrict the input variations to an extent such that the state variables are the only factor for inducing toggles. This is achievable by sustaining the same vector at the input pins over multiple clock cycles. Statistically in a purely random scenario each new vector will have at least half of the input bits toggled from the previous vector. These toggles will propagate through the transitive fanout cones of the respective inputs to generate further toggles in the circuit. If we ensure not to create any toggles at the input itself, it helps us reduce the circuit activity to a good extent because in such situation the state bits are the only factor for generating activity in the circuit. Moreover, we prefer scenarios where fewer state bits change as we keep the input vector at a stable value. Additionally, it also helps in reducing the synergistic transitions. That is, there are gates in the circuit which derives its inputs from the state-bits as well as from the inputs and they transition when more than a single input changes. If the changing state is the only dynamic variable during the operational mode, chances are less that such synergistic transitions will occur as compared with the random scenarios. Naturally these help in minimizing the overall circuit activity and keep the power consumption of the overall circuit low, which is our primary objective. We note that without sustaining a vector, the power consumption generally is much higher, closer to consuming an average power level of the circuit. The concept of *Toggle Minimization* by sustaining a vector is shown in Figures 1 (a) and (b).

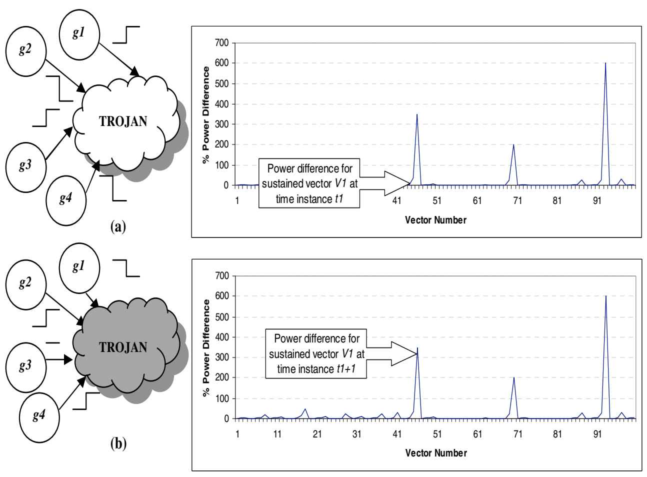

Fig. 2. Power differential measurement between vectors to isolate the gates connected to the Trojan

In our experiments, we generated a set of 1000 random input vectors, each of which is sustained to a maximum of 25 cycles. For a vector *V*, after sustaining it k times ($k < 25$), if we find that the system has reached a stable state where no further change in the state variables occurs, we move on to the next vector. Thus, a vector set for any particular circuit contains a maximum of 25000 input sequences. While holding the same vector at its input helps the circuit to traverse a local state space, changing the input vector to sustain next serves as a jump to explore some other regions of the state space. We apply this test sequence to both the genuine and the Trojan circuits in our experiments and obtain the *differential power* numbers (expressed in %) between them. The resultant plot is the *Differential Power Profile Plot* for the CUT.

B. Step 2: Infected Region Isolation

We use the *Differential Power Profile Plot* information from Step 1 to identify the region(s) of the circuit that are likely to be insertion points of the Trojan. We focus on vector pairs that produced high *differential power* as starting points.

Let us consider a sustained vector V_1 with which the CUTs shows a noticeable difference in the power profile in simulation cycles *t* and *t+1*. This is shown in Figure 2. Let g_1 to g_4 be the four internal gates in the circuit that are actually connected to the Trojan. Since the Trojan derives its inputs from the gates in its transitive fanin cone, any activity produced in the Trojan implies activity in its inputs. From the lower portion of the figure it shows that as the gates g_1, g_2, g_3 and g_4 underwent transition from 1010 to 0111, there is an observable difference in the *differential power* in the circuit. Thus a transition in gates g_1, g_2 and g_4 gives rise to a significant increase in the differential power ratio between the Trojan circuit and the genuine circuit, marking these gates as a potential suspect for the Trojan implantation.

We keep two counters for each gate: *TrojanCount* and *NonTrojanCount*. Each time a gate toggles by a vector pair

Algorithm 1 Isolate gate(s) accountable for Trojan activity

```
Require: GenuineCkt, TrojanCkt, InputVector
Ensure: Plot of gate weights corresponding to their probability of Trojan association
 1: $PowerDifferentialThreshold \Leftarrow 5.0$
 2: $TrojanCount \Leftarrow 0$
 3: $NonTrojanCount \Leftarrow 0$
 4: $ToggledGateList \Leftarrow Simulate(GenuineCkt, InputVector)$
 5: $PowerNumbers(GenuineCkt) = PowerSimulate(GenuineCkt, InputVector)$
 6: $PowerNumbers(TrojanCkt) = PowerSimulate(TrojanCkt, (InputVector))$
 7: for all $(V_i, V_{i+1}) \epsilon InputVector$ do
 8:   $\%PowerDifferential \Leftarrow (\mathbf{abs}\frac{(PowerNumbers(TrojanCkt,(V_i,V_{i+1})-PowerNumbers(GenuineCkt,(Vi,V_{i+1}))))}{PowerNumbers((GenuineCkt),(V_i,V_{i+1}))} * 100)$
 9:   if $\%PowerDifferential > PowerDifferentialThreshold$ then
10:     $IncrementWeight(TrojanCount, ToggledGateList(V_i, Vi+1))$
11:   else
12:     $IncrementWeight(NonTrojanCount, ToggledGateList(V_i, Vi+1)$
13:   end if
14: end for
15: $BuildPowerProfilePlots(PowerNumbers(GenuineCkt), PowerNumbers(TrojanCkt))$
16: for all $g_i \epsilon GenuineCkt$ do
17:   $GateWeight = \frac{TrojanCount(g_i)}{NonTrojanCount(g_i}$
18: end for
```

TABLE I
FUNCTIONS OF ALGORITHM 1

Function	Purpose
Simulate(*Ckt*, *Vector*)	Simulate *Vector* set on given *Ckt*
PowerSimulate(*Ckt*, *Vector*)	Simulate *vector* set on given *Ckt* to compute the power numbers
BuildPowerProfilePlots(PowerNumbers(*Ckt1*), PowerNumbers(*Ckt2*))	Plot the differential power between *Ckt1* and *Ckt2*
IncrementWeight(*CountArray*, ToggledGateList(V_i, V_{i+1})	Increment weights of all gates that toggled between V_i and V_{i+1} in *CountArray*

that shows *differential power* greater than the *PowerDifferentialThreshold* (which is set to 5%, a typical value of process variation [1]), its *TrojanCount* is incremented and vice versa. After this analysis is over we compute the ratio *TrojanCount/NonTrojanCount* which is called as *gate weight*. A high value of the *gate weight* indicates that the gate(s) are most likely to be associated with the Trojan.

The entire procedure is outlined in Algorithm 1 and the functions used in the algorithm are explained in Table I. *PowerDifferentialThreshold* is the maximum differential power accounted for process variation. The two counters *TrojanCount* and *NonTrojanCount* are as described above. Evidently, a highly active gate which is not associated with the Trojan will most likely toggle between vectors when power differential is above the threshold. In that case its *TrojanCount* will increase as per our algorithm. To compensate for such spurious increments, we keep the *NonTrojanCount* which increments the count of a gate which toggles when the *Differential Power* is below *PowerDifferentialThreshold*. Whenever the number of points above the threshold is less than the points below the threshold for a given gate, the ratio of these two parameters would turn out to be small, thereby filtering out highly active signals that may not be associated with the Trojan.

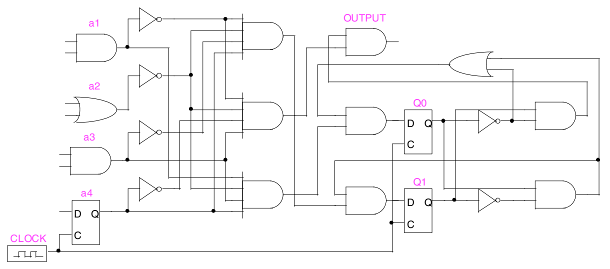

Fig. 3. Example of a Trojan circuit

IV. Trojan Description

Typical Trojan structures used in our experimental setup is shown in Figure 3, and its size is less than 1% of the original circuit for large circuits. For small circuits, we keep it less than 3%. The sequence that the Trojan attempts to detect is **1011, 0001 and 0010** in this order. The Trojan inputs $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$ and $\mathbf{a_4}$ as well as the Trojan output referred as **OUTPUT** is connected to internal nets. Table II shows the sizes of the Trojans in terms of percentage gate counts used in our experiments.

We also insert the Trojan to different locations in the original circuit. We classify the gates in the circuit into three different classes depending on their activity. To compute the

TABLE II
TROJAN SIZE (% OF TOTAL GATE COUNT)

Circuit	High-activity	Medium-activity	Low-activity
s1196	2.61	2.96	2.61
s5378	0.66	0.56	0.56
s9234	0.29	0.32	0.36
s15850	0.19	0.20	0.18
s38584	0.09	0.09	0.10

TABLE III
AVERAGE % ACTIVITY OF GATES ASSOCIATED WITH TROJAN

Circuit	High-activity	Medium-activity	Low-activity
s1196	40.4	18.8	5.3
s5378	48.2	37.4	10.6
s9234	49.7	25.5	11.6
s15850	37.7	27.7	9.9
s38584	41.8	31.7	0.4

activity of a gate in the circuit we apply a sequence of 10000 non-sustained random vectors to the circuit and then compute the toggle count of each gate. Each signal is classified as High-activity, Medium-activity or Low-activity. After classifying the signals, we embed Trojans with the groups identified. The inputs to a Trojan can come from any of the three categories. Table III shows the average percentage activity at the inputs of the Trojans for our setup.

In order to make sure that the Trojan is really stealthy we choose a triggering sequence which is very rarely occurring within the set of signals chosen. To make it undetectable we connect the Trojan output(s) to an internal gate for which the non-triggered value at the output(s) of the Trojan is non-controlling for the gate to which it is input. For instance, if the Trojan produces an output of 0 under normal operating conditions, this net is connected to an OR/NOR gate so that it does not produce any effect unless the Trojan is triggered. Further, in Step 1 of the algorithm, we check that even when the Trojan is triggered, the effect does not reach any primary output, which otherwise would be a functionally detectable Trojan. This is possible because any controlling value in the propagation path of the gate affected by the Trojan output would mask its effect.

V. EXPERIMENTAL RESULTS

The results are presented in two parts corresponding to the two steps in our methodology. Table IV shows the maximum *percentage power differential* between the genuine and Trojan circuits obtained from both random vectors and vectors generated by our approach. We embedded Trojans three different ways as discussed before: Trojans monitoring high-activity signals, medium-activity signals, and low-activity signals, respectively. *Rnd* and *Ours* used in the table stand for *Random Approach* and *Our Approach* respectively. It is evident that for nearly all cases our approach enhances the *percentage differential power*. For example, for both s1196 and s5378, more than an order of magnitude improvement was achieved using our approach when compared with random vectors. The enhanced power differential helps us to isolate the

TABLE IV
COMPARISON OF % POWER DIFFERENTIAL BETWEEN GENUINE AND TROJAN CIRCUITS ACHIEVED BY RANDOM AND OUR APPROACH

Trojan Type	High-activity		Medium-activity		Low-activity	
Circuit	Rnd	Ours	Rnd	Ours	Rnd	Ours
s1196	10.71	**300**	27.08	**650**	18.75	**600**
s5378	4.18	**133.33**	2.24	**18.18**	2.79	**7.46**
s9234	30.77	**50**	30.77	**50**	22.22	**66.66**
s15850	13.04	**38.7**	**10.07**	9.46	3.28	**5.68**
s38584	0.74	**4.62**	0.8	**1.66**	0.68	**6.52**

region where the Trojan may be embedded, which is discussed next.

The results of the second part of our analysis is shown in Figures 4, 5, 6, 7 and 8. The *x-axis* represents the gate numbers and the *y-axis* represents the computed *gate weight*. Recall that relative *GateWeight* is a direct indication of the probability that the gate is connected to the Trojan. In larger circuits, gates have higher count value in the *NonTrojanCount* because *PowerDifferentialThreshold* is exceeded less frequently during the application of the sustained vectors. As a result, the fraction *GateWeight* generally decreases as the circuit size increases. Nevertheless, it is the relative weights that count. As we can see from Figure 8, only a fraction of Total gates of the circuit is actually assigned a weight which means that the algorithm sieves out most of the gates from the larger circuits. The dotted circle in the figures correspond to the gates that are indeed connected to the Trojan in the actual circuit. As evident, in most cases the Trojan gates weigh out to a high value relative to other gates.

There are a few cases in which this method was not able to make a distinction. Our analysis on such cases led to two reasons for such anomalies:

1) Not all signals connected to the Trojan undergo transition when the Trojan is switching. As an example, for signals g_1, g_2, g_3 and g_4 connected to the Trojan if the triggering sequence is 1010 followed by 1111, then corresponding to every such sequence occurring during simulation g_2 and g_4 will have higher count in *TrojanCount* than g_1 and g_3. So during *GateWeight* computation, g_2 and g_4 will get more weight than g_1 and g_3, and this effect will show up in the *GateWeights* plot.
2) The FSM nature of the Trojan may be such that the internal state of the Trojan may change even if the inputs to the Trojan are stable. In such a case, if the overall circuit activity happens to be low, the *differential power* between the CUTs will be highlighted. But the toggling signals corresponding to such high differentials need not necessarily be connected to the Trojan.

We note that for nearly all the test cases, our approach was able to identify the signals responsible for the Trojan, and the anomalies mentioned above occur infrequently.

VI. CONCLUSION

We have presented a novel sustained vector methodology that is very effective in detecting the presence of a Trojan.

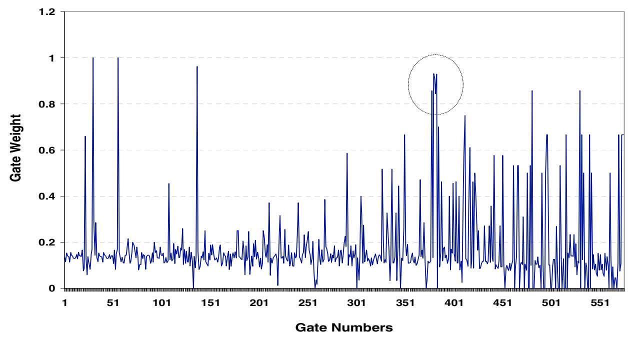

Fig. 4. Plot of Gate Weights for s1196 Medium-active Trojan

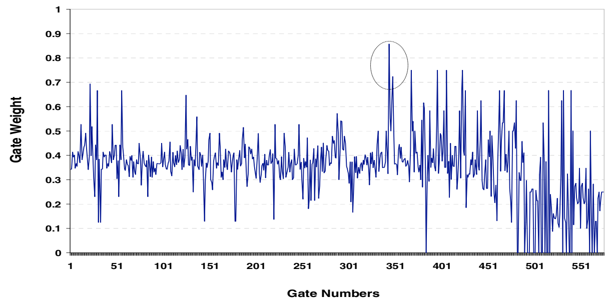

Fig. 5. Plot of Gate Weights for s1196 High-active Trojan

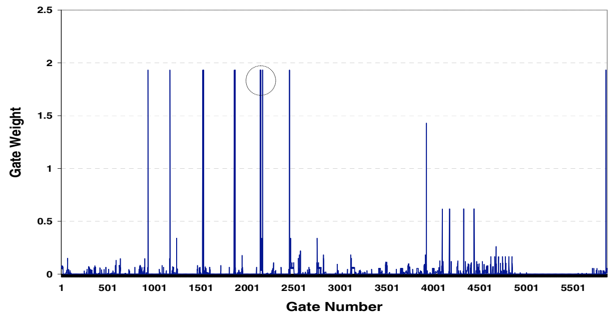

Fig. 6. Plot of Gate Weights for s9234 Low-active Trojan

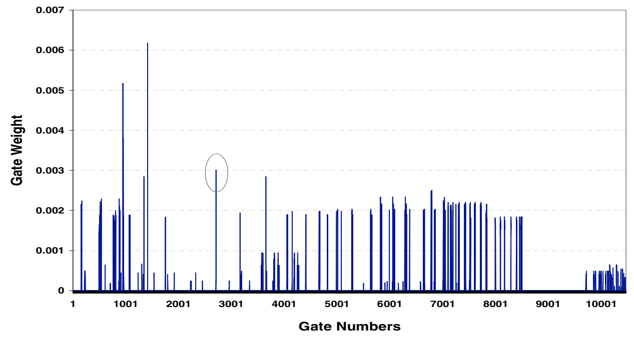

Fig. 7. Plot of Gate Weights for s15850 Low-active Trojan

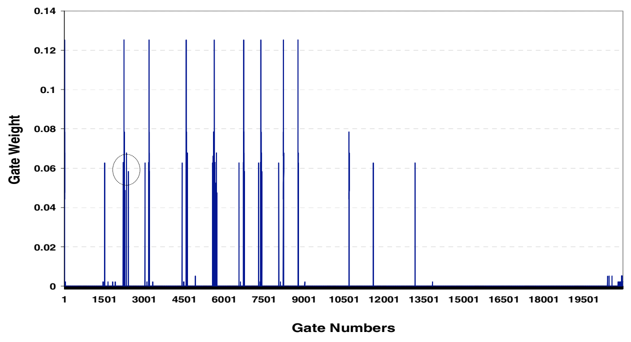

Fig. 8. Plot of Gate Weights for s38584 High-active Trojan

Even if the Trojan constitutes only a tiny fraction of the chip area, our experimental results show that this technique enhances the power profile difference between the genuine and Trojan circuits by up to more than one order of magnitude as compared with the random vectors. Furthermore, this method is effective irrespective of the activity behavior of the gates which is evident from the fact that it is successful in detecting most of the High-activity, Medium-activity and Low-activity Trojans in the benchmark circuits. Finally, in many cases, our approach was able to pinpoint the actual location of the Trojan in the circuit.

References

[1] D. Agarwal, S. Baktir, D. Karakoy, P. Rohatgi and B. Sunar; *Trojan Detection using IC Fingerprinting*, IBM Research Report, 2006.

[2] K. Nowaka, G. Carpenter, F. Gebara, J. Schaub, D. Agarwal, P. Rohatgi, W. E. Hall, S. Baktir, D. Karakoyunlu and B. Sunar; *IC Fingerprinting and Stable IS Sensors for Enhanced IC Trust*,2006.

[3] M. Banga, M. Chandrasekar, L. Fang and M. Hsiao; *Guided Test Generation for Isolation and Detection of Embedded Trojans in ICs*, ACM Great Lake Symp. on Very Large Scale Integration, 2008, pp. 363-366.

[4] M. Banga and M. Hsiao; *A Region Based Approach for the Detection of Hardware Trojans*, IEEE Int. Wkshop on Hardware-Oriented Security and Trust, 2008, pp 43-50.

[5] S. Pilli and S. Sapatnekar; *Power estimation considering statistical IC parametric variations*, ISCAS 1997, pp. 1524 - 1527, vol.3.

[6] X. Wang, M. Tehranipoor and J. Plusquellic; *Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions*, IEEE Int. Wkshop on Hardware-Oriented Security and Trust, Jun 2008, pp 15-22.

[7] C. Fagot, O. Gascuel, P. Girard and C. Landrault; *On Calculating Efficient LFSR Seeds for Built-In Self Test*, Proc. Of European Test Wkshop, 1999, pp 7-14.

[8] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski; *Logic BIST for large industrial designs: real issues and case studies*, ITC, 1999, pp. 358-367.

[9] W. T. Cheng, M. Sharma, T. Rinderknecht and C. Hill; *Signature Based Diagnosis for Logic BIST*, ITC 2006, Oct. 2006, pp. 1 - 9.

[10] C. H. Kim and J. J. Quisquater, *How can we overcome both side channel analysis and fault attacks on RSA-CRT?*, Wkshop on Fault Diagnosis and Tolerance in Cryptography, 2007, pp. 21 - 29.

[11] J. Guajardo, S. S. Kumar, G.-J. Schrijen and P. Tuyls, *Physical Unclonable Functions and Public-Key Crypto for FPGA IP Protection*, Int. Conf. on Field Programmable Logic and Applications, Aug. 2007, pp. 189 - 195.

[12] B. Gassend, D. Clarke, M. van Dijk and S. Devadas, *Controlled physical random functions*, 18th Annual Proceedings of Computer Security Applications Conf., Dec. 2002, pp. 149 - 160.

[13] http://cp.literature.agilent.com/litweb/pdf/5988-0687EN.pdf