Golden Chip-Free Detection of Automatically Inserted Hardware Trojans

Shuo Yang, Jonathan Cruz, Tamzidul Hoque, and Swarup Bhunia
Department of ECE, University of Florida
Email: {$sy, jonc205, thoque, swarup$}@ufl.edu

***Abstract*—Malicious modification of integrated circuits in untrusted design house or foundry has emerged as a major security concern. Such malicious alterations are referred as Hardware Trojans. Even though various hardware Trojan detection techniques have been proposed, their effectiveness is typically evaluated for a limited set of Trojan inserted benchmarks. To understand the true potential of a solution and to discover possible weaknesses, diverse possible implementations of various classes of Trojans should be used during evaluation. In this work, we implement a tool-based Trojan insertion technique for gate-level netlists that can automatically insert a maximum number of valid Trojans with unique configurations. Through our hardware demonstration, we would like to evaluate our golden chip-free side-channel analysis based Trojan detection approaches against the newly developed tool. The proposed hardware demonstration includes a step-by-step execution of the Trojan insertion tool and the corresponding side-channel analysis based detection process. The goal is to show how the automatic generation of a large number of Trojan instances could be leveraged for establishing confidence over any Trojan detection solution.**

I. Introduction

Malicious modifications of integrated circuits (IC) at untrusted foundry has become a serious security concern since most design houses outsource the manufacturing process. Such malicious hardware modifications, also referred to as hardware Trojans could enable covert channels or back doors through which sensitive information such as cryptographic keys can be leaked, or simply cause malfunction under certain rare conditions.

Researchers have developed various hardware Trojan detection techniques, each having their own advantages and limitations. While there is no silver bullet solution, most of the existing techniques can be used as complimentary detection mechanisms due to their unique coverage for particular Trojan models. These solutions could be broadly classified into two common approaches: 1) Logic testing and 2) Side-channel analysis. A big disadvantage of most side-channel approaches is the requirement of a golden IC. Procuring a golden chip may require destructive reverse engineering which may not be feasible for many cases. Besides side-channel approaches suffer from reduced sensitivity due to process and environmental variations. To mitigate this issue, we proposed a novel self-referencing based side-channel analysis approach which refers to comparing a chip's current signature with its own. We have explored self-referencing in both time [1] and space [2].

In temporal self-referencing or TeSR, the current signature of a chip is compared with its own in two different time windows [1]. This proposed approach can eliminate the effects of both die-to-die and within-die process variations, as well as local noise induced by other design marginalities. It also avoids the requirement of a reference or golden IC to isolate Trojan effects by comparing a chips transient current signature with itself– but at a different time window. Spatial self-referencing has also been developed to eliminate the requirement of a golden chip and the impact of process-induced variations. This technique takes advantage of the adjacent self-similar components within a chip. By comparing the current signature of identical components we effectively eliminate inter-die and intra-die (systematic) variation. Intra-die random variations can also be eliminated by considering switching of a large number of gates [2].

Simulation and experimental analysis of a detection mechanism is a primary requirement for understanding the effectiveness of a solution. To evaluate future Trojan detection techniques in a comprehensive manner, we have developed a framework to automatically and dynamically insert various Trojan types into a flattened gate-level design based on user-provided configurations. Existing trust benchmarks suffer from several deficiencies that can hamper the effort for effective comparisons and accurate results. With our framework, users can configure the Trojan structure (e.g., combinational, sequential, etc.), the number of Trojans, the number and rarity of trigger points, the payload signals, and payload effects. To account for future Trojan designs, the framework allows users to insert a *Template Trojan*, or custom Trojan structure with a chosen payload and triggering condition. In our hardware demonstration, the tool inserted Trojan infected designs are mapped onto the FPGA and the two self-referencing approaches are applied to the captured side-channel signatures.

II. Hardware Demonstration Setup

The goal of the technical demonstration is to show how quality Trojan data could provide valuable insight regarding the capabilities of any Trojan detection mechanism. Our hardware demonstration of Trojan detection has three major aspects: 1) Trojan insertion, 2) Side-channel data extraction, and 3) Decision making. Below we discuss these segments in detail.

A. Trojan Insertion

We employ the software tool described in Section I to insert Trojans into a target IP. For each IP, functional simulation is used to estimate the 0,1 signal probabilities (sp_0, sp_1) for each signal in the IP. Rare nodes are identified as signals with

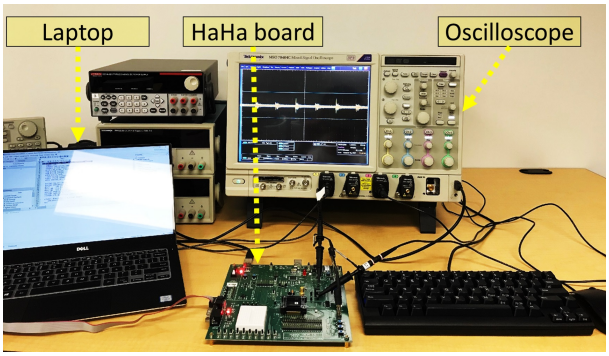

Fig. 1. A software tool running on the laptop generates Trojan inserted designs that are mapped onto the HaHa board's FPGA. An oscilloscope is used for observing the power consumption. The extracted power data is analyzed using MATLAB running on the laptop.

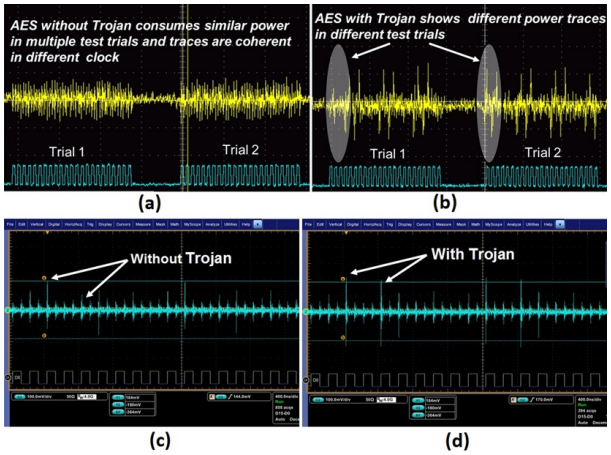

Fig. 2. Figure (a) and (b) shows the current measurement from an AES crypto module where traces are different in two test trials when the sequential Trojan is activated. Figure (c) and (d) shows the deflection of current in identical multiplier block of an ALU when the Trojan is activated.

a probability $\{sp_0|sp_1\} \leq \theta$ where θ is the user-provided threshold signal probability value. Potential Trojan instances are generated using the principle of random sampling from the population of rare nodes (and non-rare if specified). From the population of potential Trojans, both trigger conditions and payloads are verified. This process removes any false trigger conditions or unobservable payloads producing a list of feasible Trojans. Trojans are randomly selected from this list, constructed, and inserted into the gate-level IP before synthesizing onto the FPGA. By automating this process, we hope to remove any potential bias and create a red-team blue-team scenario for evaluating our Trojan detection technique.

B. Side-Channel Data Extraction

As shown in Fig. 1, to do the Trojan detection, a Hardware Hacking Security Education Platform (HaHa SEP) will be used and Trojans will be inserted in the designs which are implemented in the FPGA of the HaHa SEP. On HaHa SEP, there is a current sensing resistor that is mounted between the voltage regulator and the FPGA. Therefore, all the current that is consumed by the FPGA will go through the resistor. Thus, the side channel properties of the FPGA can be extracted by measuring the voltage drop across the current sensing resistor. A computer is needed to program the FPGA and an oscilloscope will be used to measure and record the power consumption traces. The computer will obtain the data from the oscilloscope and analyze using the self-referencing method to decide if there is a Trojan.

Figure 2 shows example measurements of both spatial and temporal self-referencing. For temporal self-referencing, the design will be working under the same set of input patterns and keep repeating the same sequence of the states. The identical sequence of the states will be included in individual time windows. The oscilloscope will be used to record multiple power consumption traces for different time windows. As the design is repeating exactly the same instructions, the power consumption for each time window should be similar for all. However, if there is a sequential Trojan triggered, which most probably has a different sequence of states, under the same input patterns, the power consumption traces for different time windows would differ (Figure 2 (a), (b)).

For spatial self-referencing, self-similar parts of a design will be activated only one at a time. When each of the self-similar parts are working, the power consumption trace will be recorded. Therefore, each of the parts will have a corresponding power trace. When none of these self-similar parts of the design have a Trojan triggered, the power consumption traces should be similar to each other and the noise and fabrication variety factors can be canceled out. However, if a Trojan experiences full or even partial trigger, power traces of the identical modules will be dissimilar (Figure 2 (c), (d)).

C. Decision Making

Once the two sets of self-similar current measurements for a given chip are collected, it is analyzed using different metrics to comment regarding the presence or absence of Trojans in that chip. The temporal and spatial self-referencing analysis is done in both time domain and frequency domain of the current consumption data. For time domain analysis, no further processing of the data is required except averaging. For the purpose of frequency domain analysis, Fast Fourier transform (FFT) is executed. Our initial experiment reveals that impact of certain Trojan models is observed more in one domain compared to the other. Based on the dissimilarities observed within the self-similar measurements a final decision is made. The whole process of data conversion, analysis, and decision making is executed automatically using a script in MATLAB.

REFERENCES

[1] T. Hoque, et al., "Golden-free hardware Trojan detection with high sensitivity under process noise," *JETTA*, 2017.
[2] Du D, et al., "Self-referencing: A scalable side-channel approach for hardware Trojan detection," *CHES*, Springer, Berlin, Heidelberg, 2010.