Implementing Hardware Trojans:

Experiences from a Hardware Trojan Challenge

Georg T. Becker, Ashwin Lakshminarasimhan, Lang Lin,
Sudheendra Srivathsa, Vikram B. Suresh, and Wayne Burelson
University of Massachusetts at Amherst, USA
{becker, ashwin, llin, ssrivathsa, vsuresh, burleson}@ecs.umass.edu

Abstract—**Hardware Trojans have become a growing concern in the design of secure integrated circuits. In this work, we present a set of novel hardware Trojans aimed at evading detection methods, designed as part of the CSAW Embedded System Challenge 2010. We introduced and implemented unique Trojans based on side-channel analysis that leak the secret key in the reference encryption algorithm. These side-channel-based Trojans do not impact the functionality of the design to minimize the possibility of detection. We have demonstrated the statistical analysis approach to attack such Trojans. Besides, we introduced Trojans that modify either the functional behavior or the electrical characteristics of the reference design. Novel techniques such as a Trojan draining the battery of a device do not have an immediate impact and hence avoid detection, but affect the long term reliability of the system.**

Keywords- Hardware Trojan, side-channel, hardware security

I. Introduction

Hardware Trojans are malicious circuits that can impact the functionality and trustworthiness of electronic systems. They can be classified based on different criteria such as the design phase in which they are inserted, the activation mechanisms required to trigger the Trojans and the effects they have on the design [1]. Implementation of hardware Trojans can be trigger-activated combinational or sequential circuits inserted at different abstraction levels. Alternatively, they can be always active but only leak out secret information through covert side channels or can be triggered by a rare event to change the functionality of the design [3][4]. A minute change in process parameters during fabrication can also act as Trojans and impact the lifetime reliability of the design [5]. The growing concern of Trojans is found in many applications. While hardware Trojans due to un-trusted foundries can have significant economic implications, their presence in medical devices can lead to life threatening situations.

Although designed to be lightweight or rarely-triggered, hardware Trojans can impact the design parameters like performance, power (both static and dynamic) and area to enable detections. Existing Trojan detection techniques can be broadly classified based on a) logic testing or b) side-channel analysis. Logic testing includes equivalence checking at the pre-silicon design phase and generation of specific test patterns through Automatic Test Pattern Generation (ATPG) to excite critical paths during chip testing [6]. With increasing design complexity, logic testing may not be feasible for multi-million gate designs. The insertion of Trojan provides side-channel in the form of variations in operating frequency or power, which can be utilized to detect the trustworthiness of the design. In [7], the path delays of a set of genuine chips are used to create a library of delay fingerprints, incorporating the effect of process variations. The path delays of the rest of the suspected chips are compared to the golden database to detect any form of Trojans. A similar approach of fingerprinting both the chip frequency and the dynamic current [8] is proposed as a non-invasive side-channel analysis based detection technique. Leakage current as another type of side-channels can also detect Trojans with a high probability [9].

With the advanced process technologies, the major challenge of Trojan detection is to increase the detection sensitivity of parametric variations and to build accurate statistical models to differentiate the effects of process variations from those of Trojans. In this paper, we present the hardware Trojans we implemented during the CSAW Embedded Security Challenge 2010. The challenge included two parts. In the first part, the objective was to extract the secret key from the Alpha design. The second part aimed at Trojan insertion to modify the behavior of the design. The rest of the paper is structured as follows. Section II introduces each of our five hardware Trojans for the Alpha challenge, in which we either reveal the secret key or the plaintext of the Tiny Encryption Algorithm design. The Beta Trojans which modify the functionality of the given 4-bit adder reference design are discussed in Section III. We conclude in Section IV with some results and a short discussion why our Trojan could be detected by the used Trojan detection methods in this challenge.

II. Alpha Trojans

The alpha design implements the Tiny Encryption Algorithm (TEA) and is hardened with a Trojan detection mechanism that tries to find suspicious changes in the delay of a critical path. Details of how exactly this delay-based detection method works and what part it protected was not given to the participants. The goal of the alpha Trojans is to recover the secret key or the used plaintext while defeating the detection mechanism. Our alpha Trojans can be divided into two categories: side-channel Trojans and triggered Trojans. The first two Trojans will use side-channels to secretly reveal

This work was supported by the NSF Grants 0916854, 0964641, 0923313, NSF travel grant CNS 0958510 and Xilinx.

the key. These Trojans do not change the functionality of the design and do not use a trigger mechanism as they are always active. Trojans number 3, 4 and 5 on the other hand are trigger based Trojans that will only reveal the key or plaintext once they are activated.

A. Trojan 1: Spread Spectrum based Side-channel Trojan

In this Trojan we use the EM side-channel to covertly transmit the secret key. The used technique is very similar to classic CDMA communication techniques. The main idea of this type of Trojan was introduced in [3] using the power side-channel. The Trojan consists of two parts, a signal generating circuit that encodes the key and a leakage generating circuit that leaks out (transmits) this bit stream. The signal generating circuit itself consists again of two parts, a MUX that selects a key bit to be transmitted and a 10 bit LFSR that generates a 1023-bit long bit-stream. The output of the signal generating circuit is the exclusive-or sum of the selected key bit and the output bits of the LFSR.

The function of the leakage generating circuit is to create an EM-signal with high amplitude when the output of the signal generating circuit is a logical "1" and a signal with low amplitude when the output is a "0". This can be realized in many different ways. In our design we used several ring oscillators with an enable signal. The ring-oscillators are only active when the input signal is "1" and inactive when the input is "0". This will result in a higher switching activity when the output is "1" and hence also in a higher EM signal. One idea behind using a ring-oscillator for the leakage generating circuit is that the frequency of the ring-oscillator is different from the clock frequency. Using signal processing this should result in a good signal-to-noise ratio.

To derive the key we use a side-channel analysis. At first we collect an EM trace of the FPGA while it is running. To collect measurements we used a DPO7014 oscilloscope from Tektronix and a 1cm loop H-field probe from the ETS-Lindgren near field probe set. We then compute the expected 1023 bit long bit-stream of the used LFSR. In the next step we need to map this bit-stream to our measurement data. As the LFSR is clocked once per clock-cycle, we need to map one bit of the LFSR output to 20 measurement points. We simply do this by copying each output bit of the LFSR 20 times and storing the result in an array. In this way we get a 1023bit * 20 = 20460 bit long hypothesis for one complete iteration of the LFSR. We copy this bit-stream until the length of the bit-stream matches the length of our trace. In the last step we perform a cross-correlation of this hypothesis with the trace. A correlation peak should be visible when he LFSR sequence is aligned with the LFSR in the trace. If the currently selected key bit is "0" this correlation peak is positive while it is negative if the key bit is "1". This method is well known in communication systems as CDMA. We transmit one bit at a time in this way. In our design the next key bit was selected using an input button, but other methods such as a counter could be used as well.

To make the verification of our Trojan easier for the CSAW competition we connected the ring-oscillator to an output pin. As the wire of an output pin generates a very strong EM signal this drastically increases the SNR of the signal. Furthermore, it made sure that the ring-oscillators were not removed during synthesis. Our experimental results of this Trojan can be seen in Figure 1. A positive correlation peak in the left picture shows the transmission of a "1" and the negative correlation peak on the right indicates that the transmitted key bit is "0".

Of course, in practice the use of output pins should be avoided. To show that this type of Trojan can work without the use of any pins we would like to refer to [3] and [2]. In [3] simulation results were presented for this type of Trojan using the power side-channel. In [2] a side-channel hardware watermark was introduced and evaluated on an FPGA that uses the same method to secretly transmit an ID. Although both papers focus on power analysis, the EM side-channel could have been used as well.

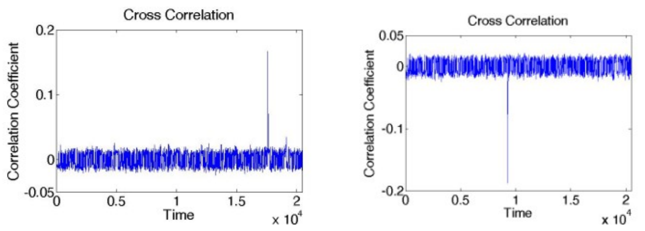

Figure 1: The cross-correlation of the LFSR-bit-stream with the EM-trace. The positive correlation peak in the left shows that the key-bit in this measurement is 0. In the right picture the keybit was set to 1 which is indicated by a negative correlation peak.

B. Trojan 2: Input-Modulated Side-channel Trojan

Similar to the previous Trojan, in this Trojan we leak out the key using a side-channel. But this time the key is leaked out over many different iterations of the program, rather than over time as it has been the case with the previous Trojan.

The Trojan consists of two components: a combination function and a leakage generating circuit. The combination function takes eight input bits and eight bits of the key to generate a one bit output. In our implementation we used a simple combination function that ANDs eight key-bits with eight input-bits and then computes the XOR sum of these values (see Figure 2).

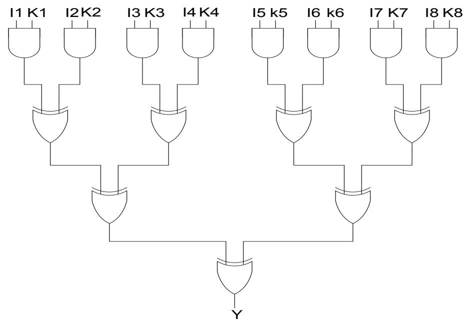

Figure 2: Combinational logic block of the Trojan

This output bit Y is then leaked out ("transmitted") using the leakage generating circuit. The function of the leakage generating circuit is to generate a high EM radiation when the output of the combination function is '1' and less EM radiation when the output is '0'. We realized this using 10-bit shift registers initialized with alternating 1s and 0s that are clocked in case the output is 1 and are not clocked if the output is 0. But to make the verification of the Trojan easier we again connect the shift register to an output pin. We want to refer again to [3] and [2] to show that it is possible to use this type of Trojan without artificially amplifying the signal. The use of the output pin was only chosen to make the detection of the Trojan easier in the CSAW challenge and due to the tight timeframe of the challenge.

The result of this Trojan is a key and input depended EM radiation whose amplitude is chosen by the leakage generating circuit. This enables us to perform a differential EM analysis to reveal the key. To do this at first we took EM-traces of 10.000 measurements for different inputs. In the next step we aligned the traces and then performed a correlation analysis to reveal the correct key. This is done by first computing the output of the combination function with the input of each trace and all possible eight-bit sub-keys. Each key guess represents a separate hypothesis which gives us 256 different hypotheses. In the last step we correlated each of these hypotheses with our measured traces. The correct key hypothesis should generate the highest correlation as this hypothesis is the best prediction of the EM radiation of the leakage generating circuit. This setup can leak out eight bits of the key at once. To be able to leak out the entire 128 bits of the key we subsequently add all of the 16 key bytes to this combination function. At different times we can therefore detect the different key bytes. Hence, using traditional side-channel analysis methods we can recover the key, even if the original design is resistant against side-channel attacks.

The main advantages of the introduced side-channel Trojans are that they 1) do not need any trigger mechanism and 2) that they do not modify the functionality of the design in any way. Hence, this type of Trojan should not be detectable using functional testing. Furthermore, this Trojan only reads from register values but does not insert any logic gates into any functional path. Therefore these types of Trojans have the potential to have minimal impact on the delay path. Taking other detection besides delay-based detection into account, the Trojan might be detected by comparing or analyzing the power consumption as the inserted gates as well as the leakage generator will add to the overall power consumption. We would like to point out again, that the leakage generator can be very small if the attacker prepares a clean measurement setup or is willing to take a lot of measurements..

The difference between the two Trojans is that in the spread spectrum Trojan you only take one long measurement while in the input-modulated Trojan you take a lot of small measurements. Both approaches have advantages and disadvantages. If only a few measurements are possible, e.g. because the number of executions of the program is restricted, the first method is most suited. The input modulated Trojan on the other hand has the advantage that if the attacker collects a lot of traces, e.g. more than a million, he can detect the Trojan even if the signal-to-noise ratio of the Trojan is very small and the signal is hidden deep in the noise.

C. *Trojan 3-5: Trigger-Based Alpha Trojans*

The other three Trojans implemented for the Alpha challenge were Trojans using a trigger mechanism. The first Trojan was triggered (activated) by a specific input pattern and the Trojan would then display the key directly on the LCD. In this Trojan we only inserted gates into the functional path that selects the LED output. The details for the delay-based detection method used in the alpha challenge were not given and our hope was that only the encryption function and not the LED output was protected. As we do not add any gates into any critical path of the encryption function we hoped that the introduced delay (due to additional capacitances for additional wires and different routing) would be small enough to avoid detection.

The fourth Trojan was again a Trojan using a trigger mechanism. This time we did not use a specific input pattern to activate the Trojan but measured the time-difference between inputs. The idea behind this was that it is very unlikely that in a functional testing such specific delay would be tested. To output the key we used output pins that were not used in the original design. By using part of the board that was not used by the Trojan-free design we avoided to add any gates into any path of the original design. By doing so we hoped to minimize the introduced delay of our Trojan.

In the last Trojan for the alpha challenge, we triggered the Trojan after a specific event elapsing x-times. In our implementation, we triggered the Trojan after the reset button was pushed x-times. The reset button is not part of any critical path and therefore this activation method should not have impact on the delay-based detection method for the encryption core. To output the Trojan we again used output pins that were not used in the Trojan-free design.

III. BETA TROJANS

Our aim in this scenario is to create and insert Trojans in the 4-bit adder Beta design in such a manner that it is not detected by the ring-oscillator based beta hardening mechanism. The Trojans should do some damage to the board, interrupt the normal functioning of the board or display the wrong answers. To avoid the beta detection mechanism we tried to insert our Trojan circuits at locations that are not protected by this hardening technique. Hence we do not directly modify the logic of the 4-bit adder which serves as a reference design but will either modify the inputs before they reach the 4-bit adder or we will directly modify the output of the 4-bit adder. Each of our Trojans in the Beta challenge consisted of two parts, a trigger mechanism and a malicious function.

We used the same trigger mechanism as the ones from the Alpha challenge. Namely, one Trojan was triggered at a specific sequence of inputs. For another Trojan we again counted the number of resets and triggered after a certain number of resets occurred. We furthermore used again a timing activated Trojan that was triggered when a specific delay between inputs occured. Our hope was that these trigger mechanisms would not have an impact on the detection method as they were inserted either before the protected part of the design in the case of the input and delay based trigger mechanism) or in case of the reset activated Trojan using an input not part of the encryption core.

The second part of each Trojan was the malicious function that was executed once the Trojan was triggered. Our goal was again to insert this malicious function into a part of the design that was not protected by the detection method. In one Trojan we simply displayed the input at the LED instead the correct output once the Trojan was triggered. For another Trojan we implemented a denial-of-service type of Trojan that entered a state from which it would only recover after the board was powered off and on again. In another Trojan we would constantly reset the board. These three Trojans are rather typical examples of either denial-of-service type of Trojans or Trojans that produce false responses. The last Trojan we implemented for the Beta Challenge was more subtle. This Trojan drains the battery once it is activated. We did this by using LUTs not needed in the original design as shift registers and constantly shifting them. This results in a huge increase of the power consumption of the board and hence decreases the expected lifetime of the battery significantly. This type of Trojan is more subtle as its effects are not visible immediately and recovering from it is also more difficult, as simple resetting the board will not solve the problem of an empty battery.

IV. CONCLUSIONS

In this paper, we have discussed the Trojans that we designed and implemented for the CSAW Embedded System Challenge 2010. The Trojans varied from simple triggered based Trojans to sophisticated side channel based Trojans. Although the Trojans worked as planned, they were detected by the detection methods employed around the original designs. We think the main reason for this was due to different routing and placement of the Trojan-free and the Trojan design. While we designed Trojans that could be inserted into the reference design without inserting gates into critical paths, we did not make sure that the routing and placement of the Trojan-free design and the Trojan design would be the same. But even if you insert only a few gates this can change the routing and placement of the FPGA a lot. This impacted the delay of the critical path of the alpha detection mechanism as well as the frequency of the ring-oscillators used in the beta detection mechanism. Hence, our Trojans were detected by these detection mechanisms.

Trojans such as the presented Side-channel Trojans have great potential to be undetected if they can be inserted at places that are not directly protected by a Trojan detection mechanisms. However, it needs to be made sure that the insertion of the Trojan gates do not also change the routing and placement of the protected part of the design. In FPGAs this can be a very difficult task. As to how easier it becomes if ASICs are used instead of FPGAs remains an open question. We would also like to point out that the Trojan detection mechanisms that detected the Trojans required a golden model. Meaning that a Trojan-free chip is needed as a reference to detect the Trojans. This is quite difficult in practice. For FPGA applications it would be quite easy to verify that a design is Trojan free if you have a golden model - it would be enough to simply compare the bit-streams. The detection methods showed great potential for FPGAs and therefore seem to be a good solution for ASICS as well. However, as to how far it might be easier to decrease the impact of routing and placement in an ASIC and thereby avoiding detection is still an open question.

REFERENCES

[1] R. Karri, J. Rajendran, K. Rosenfeld and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," IEEE Computer Magazine, vol.43, no.10, pp.39-46, Oct. 2010.
[2] G.T. Becker, M. Kasper, A. Moradi, and C. Paar, “Side-channel based watermarks for integrated circuits,” In IEEE International Symposium on Hardware-Oriented Security and Trust (HOST 2010), pp. 30 –35, 2010.
[3] L. Lin, M. Kasper, T. G¨uneysu, C. Paar, and W. Burleson, “Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering,” In Cryptographic Hardware and Embedded Systems (CHES), volume 5747 of LNCS, pp. 382–395, Springer, 2009.
[4] Y. Jin, N. Kupp and Y. Makris, "Experiences in Hardware Trojan design and implementation," In HOST, pp.50-57, 2009.
[5] Y. Shiyanovskii, F. Wolff, A. Rajendran, C. Papachristou, D. Weyer and W. Clay, "Process reliability based Trojans through NBTI and HCI effects," In the 2010 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), , pp.215-222, 2010.
[6] M. Banga and M. S. Hsiao, “Trusted RTL: Trojan Detection Methodology in Pre-Silicon Designs”, In IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 56-59, 2010.
[7] Y. Jin and Y. Makris, “Hardware Trojan Detection Using Path Delay Fingerprint,” In IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 51-57, 2008.
[8] S. Narasimhan et al, “Multiple-Parameter Side-Channel Analysis: A Non-Invasive Hardware Trojan Detection Approach”, in IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 13-18, 2010.
[9] J. Aarestad, D. Acharyya, R. Rad, and Jim Plusquellic, “Detecting Trojans Through Leakage Current Analysis Using Multiple Supply Pad I_{ddq}s”, in IEEE Transactions on Information Forensics and Security, Vol. 5, No. 4, pp. 893-904, 2010.