EM-Based Detection of Hardware Trojans on FPGAs

Oliver Söll*, Thomas Korak*, Michael Muehlberghuber†, and Michael Hutter*
*Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria.
†Integrated Systems Laboratory (IIS), ETH Zurich, Gloriastrasse 35, 8092 Zurich, Switzerland.
Email: o.soell@student.tugraz.at, {thomas.korak, michael.hutter}@iaik.tugraz.at, mbgh@iis.ee.ethz.ch

***Abstract*—The detectability of malicious circuitry on FPGAs with varying placement properties yet has to be investigated. The authors utilize a Xilinx Virtex-II Pro target platform in order to insert a sequential denial-of-service Trojan into an existing AES design by manipulating a Xilinx-specific, intermediate file format prior to the bitstream generation. Thereby, there is no need for an attacker to acquire access to the hardware description language representation of a potential target architecture. Using a side-channel analysis setup for electromagnetic emanation (EM) measurements, they evaluate the detectability of different Trojan designs with varying location and logic distribution properties. The authors successfully distinguish the malicious from the genuine designs and provide information on how the location and distribution properties of the Trojan logic affect its detectability. To the best of their knowledge, this has been the first practically conducted Trojan detection using localized EM measurements.**

Keywords*—Hardware Trojan injection, side-channel analysis, electromagnetic emanation, Trojan placement, RapidSmith.

I. Introduction

Trust and in particular the proof of immutability of software components are indispensable needs in computer science for many years now. Throughout the last decade, trustworthiness in hardware components got equally important as these components should provide a reliable base for their software counterparts [1]. In order to assert oneself in the economic competition, many vendors abandon in-house development and fabrication, but instead set their hopes on external building blocks, and furthermore separate design from fabrication. This outsourcing incorporates new players and causes new threats regarding trustworthiness. Especially the threat of an untrustworthy player adding additional malicious circuitry, a so-called hardware Trojan horse [2], at the fabrication or deployment phase poses a severe danger. Consequential demands for means to detect modifications of the design arise.

Since the Trojan's target is to be not apparent, the design principle is to keep its size to a minimum. Different architectural types of Trojans have been presented, fulfilling this principle. Experience has shown that in particular sequential Trojans are hard to detect, since they are on the one hand very hard to trigger, because they listen to a sequence of bits of the input which is most likely not covered by any test case and on the other hand can be implemented by only occupying very few area [3]. However, side-channel analysis put out to be a strong mean to detect these malicious components [4].

The work presented in this article has been supported by the European Cooperation in Science and Technology (COST) Action IC1204 (Trustworthy Manufacturing and Utilization of Secure Devices - TRUDEVICE) and by the Swiss Commission for Technology and Innovation (CTI) under project number 13044.1 PFES-ES.

In this work, we present techniques to place additional circuitry to an FPGA design without needing its hardware description language (HDL) representation. For that, we use RapidSmith [5], a library for low-level manipulation of partially placed-and-routed FPGA designs in order to modify an intermediate data format of a Xilinx Virtex-II Pro FPGA design. We show how to attach a sequential hardware Trojan to the IO ports of an existing Advanced Encryption Standard (AES) architecture and permute the location and the distribution of the logic gates of the Trojan in order to give statements at which position the Trojan is harder to detect using electromagnetic emanation (EM) side-channel analyses. The main part of this article will propose techniques to detect the different implementations of the Trojan by locally measuring the EM of the FPGA. We do not only aim to detect the Trojan, but also provide first suggestions at which positions on the FPGA the Trojan is harder to detect. In order to achieve these goals, we utilize an electromagnetic probe and step over the package of the FPGA measuring the electromagnetic field for each step point. By utilizing simple analysis techniques, we were able to successfully distinguish the malicious from the genuine designs based on the EM side channel.

The remainder is structured as follows. First, we give an overview of related work. Sect. II describes a typical FPGA design flow and potential attack scenarios. Sect. III presents the Trojan and the measurement setup. Sect. IV provides results and conclusions are drawn in Sect. V.

A. Previous Work

Several Trojan detection mechanisms have been proposed recently, which can basically be subdivided into destructive and non-destructive approaches. Since the former comprise rather costly techniques, significantly more research has been accomplished with regard to the latter. Non-destructive detection methods can further be distinguished in terms of the countermeasures added to a given design (architectural changes), which should simplify the detection process of a potential Trojan [6]. Side-channel analysis based detection approaches use physical characteristics such as timing [7], power consumption [4], [8], or multiple characteristics [4] to generate "hardware fingerprints", which are then used to distinguish a malicious design from a genuine one. Some authors proposed to use EM as a further parameter to their multiple-parameter side-channel detection approaches [4], [9], but to the best of our knowledge, only [10] actually accomplished a setup including EM measurements in their experiments. As opposed to [10], who used EM traces gathered from a single location on top of an FPGA, we propose to use a localized approach by stepping over the FPGAs package.

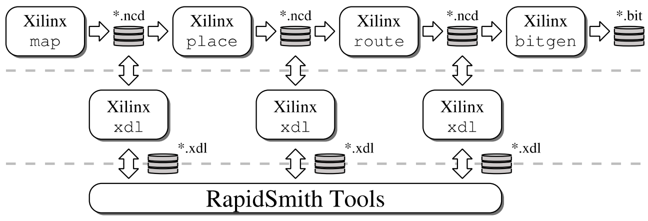


Fig. 1: Xilinx design flow and RapidSmith interface

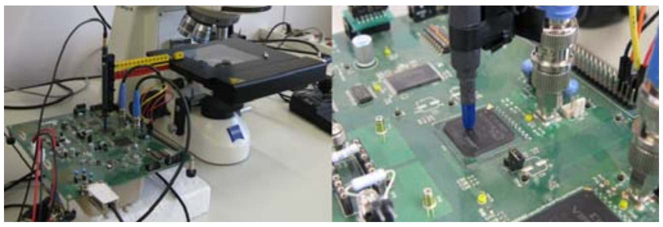


Fig. 2: Stepper table, SASEBO-G, and HF near-field probe

II. FPGA Design Flow and Attacker Model

The top row of Fig. 1 describes an FPGA design flow using Xilinx build tools, starting with the mapping of a netlist to the available, FPGA-specific resources. An attacker with access to these parts of the development chain may incorporate malicious circuitry into an existing design by modifying the intermediate *.ncd file format. This task can be simplified by employing third-party tools such as RapidSmith [5], which is a library for low-level manipulation of partially placed-and-routed FPGA designs. RapidSmith is a set of tools and APIs written in Java, which ease the import, manipulation, and export of FPGA designs. It facilitates to manipulate existing designs, add further circuitry, and export the design, by providing a low-level interface to the main components of the FPGA. It is compatible to the Xilinx Design Language (XDL), a human-readable file format equivalent to the Xilinx proprietary Netlist Circuit Description (NCD).

Fig. 1 shows that different possible entry points to the RapidSmith tools exist. For our investigations, we used RapidSmith release 0.5.2 and provided it a placed-and-routed design in *.xdl file format by converting the *.ncd file to an *.xdl file with Xilinx's `xdl` tool in advance. Although RapidSmith does not allow direct manipulation of *.bit files, an adversary may also reverse-engineer one using tools such as the *Bitfile Interpretation Library (BIL)*[1] [11] and apply a similar attack afterwards. In today's security-critical FPGA applications, the deployed bitstreams are usually encrypted. Nevertheless, several works in the recent past demonstrated that based on side-channel analyses these encrypted configurations can be retrieved for different types of FPGA platforms [12], [13].

III. Trojan Design and Measurement Setup

As a target design, we selected an AES-128 architecture. Since the Trojan should be kept small in size, we decided to implement a denial-of-service (DoS) Trojan, which required only 15 slices and hence, did not modify the original bitstream appreciably (the overall design needs 2 222 slices).

We injected the same Trojan as presented in [14]. Taking existing hardware Trojan taxonomies [15], [16], [17] into account, this Trojan can be categorized as a sequential denial-of-service Trojan placed next to the I/Os, which can be triggered externally. Such sequential Trojans are particularly hard to detect, since the kill sequence is most likely not covered by any test case and they can be implemented by only occupying very few area. The Trojan listens to two of the input signals and waits for a 30-bit kill sequence to be present. Since our data input port only has a width of eight bits, previous inputs have to be memorized. We store the incoming data in two 15-bit wide shift registers, which are connected to two of the data input bits. These flip-flops use one of the input handshaking signals, which indicates whether new data is available at the input, as clock signal. If both sequences of the two input bits match a predefined kill sequence, another input bit gets inverted.

Our measurement setup consists of the following parts: the SASEBO-G side-channel evaluation board, a digital storage oscilloscope (the LeCroy WavePro 725Zi), an EM stepper device with an EM measurement probe, and a PC running Matlab as controlling software. Fig. 2 shows the SASEBO board and the EM stepper on the left side and a zoom into the EM stepping location on the right side. The SASEBO board is connected to a PC via a serial interface. It consists of two FPGAs, one Virtex-II Pro XC2VP30 and one Virtex-II Pro XC2VP7. One FPGA can be used as a controller while the other FPGA is used to evaluate a cryptographic implementation. Both FPGAs have been clocked with a frequency of 16 MHz, supplied by a Digimess FG100 function generator.

We used the Virtex-II Pro XC2VP30 as the control FPGA and configured the Virtex-II Pro XC2VP7 with either the genuine or the Trojan-containing design. We further used a serial-to-AMBA APB bus interface to access the internal registers of the designs. The communication between the two FPGAs works as follows: First, the PC transmits 256 bits of data (the 128-bit cipherkey and a 128-bit plaintext message) over the serial interface. This data is stored in an internal register. After that, the data is transmitted to the evaluation FPGA (in chunks of eight bits where the correct order and the timing is guaranteed by a dedicated handshake protocol). After the AES calculation, the result is sent back to the control FPGA, which transfers the data back to the PC. Fig. 3 shows a schematic view of the setup and the implemented communication flow.

As an EM probe, we used a magnetic near-field probe from Langer EMV Technik (LF B3). The probe was attached to a stepper table which can be controlled by software via Matlab. The stepper allows to step over the evaluation FPGA. The accuracy (the number of step points and the step window) can be configured in software. In our experiments, we stepped over a window of about 1×1 centimeters and moved the probe in steps of $300\,\mu$m (31 steps in the x and y axis). At each step location, we measured 2 500 traces of the same operation (keeping key and message data constant) and calculated the mean trace in order to reduce noise. We therefore obtained 961 mean EM traces for each implementation. We set the sampling rate of the oscilloscope to 500 MS/s and configured

[1] At the time of writing, BIL supports only Xilinx Virtex-5 devices.

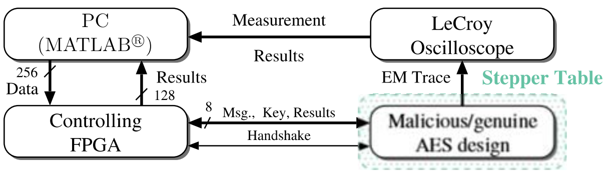


Fig. 3: Schematic view of the measurement setup

the oscilloscope to record the traces in a sequence mode. This means that 500 traces are recorded in a sequence by the oscilloscope and are transferred to the PC in sets of 500 traces which significantly improves the measurement speed.

In order to successfully locate and identify the injected Trojan on the Xilinx Virtex-II Pro XC2VP7, it is necessary to have knowledge about the layout and the individual components included in the FPGA. The XC2VP7 features an embedded PowerPC processor which can be clearly identified in the layouts (cf. major building block in Fig. 4a–Fig. 4f). The FPGA further provides 8 RocketIO transceiver blocks, 1 232 Configurable Logic Blocks (CLBs)[2], 44 dedicated 18×18 bit hardware multipliers, 44 18 kbits Block RAMs, and 4 Digital Clock Managers (DCMs). The entire Virtex-II Pro family has been fabricated in a 130 nm CMOS process technology with 9 metal layers.

We developed a Java program using the RapidSmith API, which allows us to inject the Trojan into the final FPGA layout. The program is able to place the Trojan at different locations in the design. We generated six different FPGA designs where the Trojan has been placed: 1) in the bottom-left corner of the FPGA, 2) in the top-right corner, 3) in the center, 4) distributed over the whole layout, 5) at the right side near the I/O-lines, and 6) automatically after the design has been completely re-routed by the Xilinx ISE place-and-route tool.

Fig. 4 shows the six designs and illustrates the principal layout of the Virtex-II Pro die. The PowerPC is drawn as a yellow box right in the middle of the layout. Block RAMs (BRAMs) are drawn in purple color (six vertical lines) and the Digital Signal Processing (DSP) units are drawn in orange. The clock is represented in brown (vertical line in the middle) or dark-blue (horizontal lines). Multi-gigabit transceivers and Digital Clock Managers (DCMs) are located in the top and bottom sides of the BRAMs. We further marked all required (and used) CLBs in white color and all unused CLBs are drawn in blue. For example, Fig. 4f uses different CLBs than the other designs because the design has been automatically re-routed by the Xilinx place-and-route tools. For all other designs, we told the router tool to route only those parts which have not been routed before (thus, keeping the overall layout and resource requirements nearly constant).

The resource requirements of the Trojan itself are as follows. Since in total a sequence of 30 bits (15 bits from DataInxDI[0] and 15 bits from DataInxDI[2]) has to be stored, 30 flip-flops (FFs) are needed. A Virtex-II Pro slice contains 2 FFs so we need a total amount of 15 slices to implement the Trojan. In particular, we decided that the FFs of the top-half of the slices are used to store the sequence of DataInxDI[0] and the FFs of the lower-half are used to store the sequence of DataInxDI[2]. These FFs are further connected to a shift register. For this purpose, we used the Look-up Tables (LUTs) of the available slices and implemented the logical function of a kill-sequence comparator. Because the LUTs from the top-half and lower-half of a slice have four inputs ($D1-D4$), we can compare two bits with one LUT. The first input is used for the first bit of the sequence, the second input for the first bit of the fixed kill sequence to compare with, the third input for the second bit of the sequence, and the fourth input for the second bit of the kill sequence. The comparison is done with the following logic function: $((D1 \wedge D2) \vee (\overline{D1} \wedge \overline{D2})) \wedge ((D3 \wedge D4) \vee (\overline{D3} \wedge \overline{D4}))$. All the resulting intermediate results are combined with a logic `AND`, and the final result is `XOR`ed to the DataInxDI[1], which inverts the signal if the sequence at the inputs match the kill sequence (final result equals 1). Note that in Fig. 4d, we distributed the Trojan logic on 15 different slices in 15 different CLBs. For all other designs, we tried to use all freely available slices within one CLB to inject the Trojan (needing 4 CLBs at the most).

[2]Each CLB of the XC2VP7 consists of 4 slices and two 3-state buffers where each slice consists of two 4-input LUTs and two FF registers.

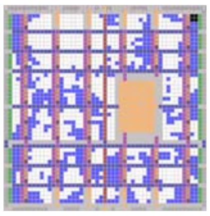
IV. Results

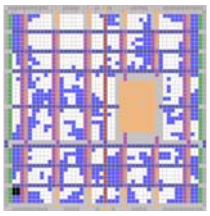
We compared the EM traces of the genuine design with six different malicious designs containing differently placed Trojans. For this purpose, we calculated the absolute difference of all measured EM traces for the 961 EM-stepping points. Furthermore, we only focused on the I/O communication part of the implemented AES core since we assume Trojan activity in that time interval.

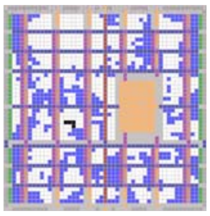
Regarding the post-processing of the traces, we applied the following techniques. First, we aligned the traces in horizontal dimension because they were not perfectly aligned because of noise and clock jitter. Second, in order to identify points of interest (i.e., points where we assume a Trojan-dependent leakage), we calculated the variance for each difference vector and considered the sample point with the highest value.

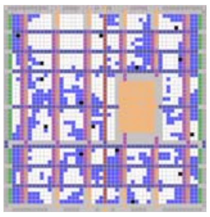
Fig. 5a to Fig. 5f depict the EM-signal differences of the genuine design and the six different malicious designs. In order to create the 2-D plots, we mapped the difference vector to a matrix with 31 rows and 31 columns. Each point represents a different EM-probe location where the color represents the difference of the mean EM traces at the point of interest (blue means almost no difference and red indicates a high variance). As a first observation, one can identify the high variance of the re-routed design. This was expected because the entire design was automatically re-routed and provides a significant difference in the EM emanation compared to the original design. Interestingly, it shows a high variance in the top-right corner where the Trojan has been placed to. Note that we measured the emanation over an area of about 1×1 centimeters, so the plot shows not only the direct EM signals of the FPGA die (assumed to be located in the middle of the plot) but also the indirect emanation from bonding wires including ground lines and I/O communication.

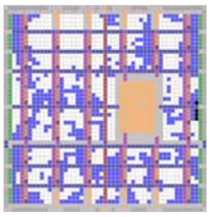
It also shows that when the Trojan has been placed in the top-right or bottom-left corner of the FPGA, the Trojan-dependent signals are higher than in cases where the Trojan has been placed in the center of the FPGA or when the Trojan


(a) Top-right


(b) Bottom-left


(c) Center


(d) Distributed


(e) Next to I/O pins

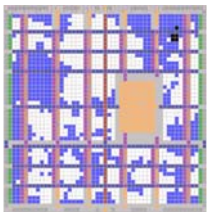
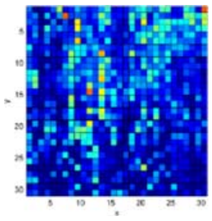
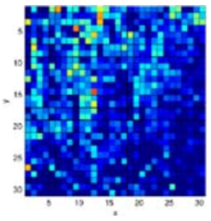
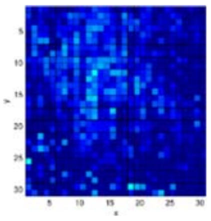
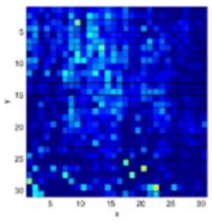

(f) Re-routed

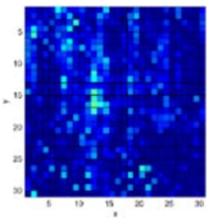
Fig. 4: The six FPGA designs with the Trojan placed at different locations (Trojan logic drawn in black)


(a) Top-right


(b) Bottom-left


(c) Center


(d) Distributed


(e) Next to I/O pins

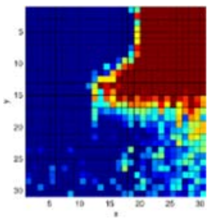

(f) Re-routed

Fig. 5: Difference between Trojan-free and malicious designs for six different Trojan-placement locations

has been distributed over the layout. The difference in Fig. 5a (top-right) and Fig. 5b (bottom-left) are higher than the signal difference in Fig. 5c (center) and Fig. 5d (distributed). The signal differences of Fig. 5e (near I/O) seem to be higher but not as significant as the results obtained as for the top-right and bottom-left cases. There are several possible reasons for that observation. One reason might be the fact that the Trojans that have been located near to VCC or ground lines (like for the top-right and bottom-left cases) have a higher influence (through EM-signal modulation or shorter wire lengths) on the power supply signals and can therefore be more easily detected. The Trojans that have been placed in the top-right and bottom-left corners are indeed close to many VCC and ground pins of the FPGA. Another reason could be that if the required CLBs of a Trojan are located right next to each other or are placed very close together, the stronger will be the signal leakage. For the top-right and bottom-left cases, CLBs have been used that are unused by the genuine design. The Trojan that has been placed in the center of the FPGA has been placed right next to CLBs that already consume a significant amount of power.

V. Conclusions

We presented first results of EM-stepping analyses of hardware Trojans that have been injected into an FPGA. We were successfully able to detect if a hardware Trojan was integrated into the design by comparing the EM emanations with a reference design. Six different FPGA designs were investigated where the Trojan logic was placed differently over the entire FPGA layout. We evaluated the efficiency of EM side-channel based Trojan detections and identified that the location of the Trojan logic plays an important role in the detectability of the malicious circuit.

As future work, we plan to use different and more fine-grained EM probes on a de-capsulated/open FPGA. Furthermore, we want to compare the efficiency of SCA-based Trojan detection on ASIC and FPGA-based designs.

References

[1] Defense Advanced Research Projects Agency (DARPA), "Trusted Integrated Circuits (TRUST)," 2007.

[2] M. Tehranipoor and B. Sunar, "Hardware Trojan Horses," in *Towards Hardware-Intrinsic Security*, A.-R. Sadeghi and D. Naccache, Eds. Springer, 2010, pp. 167–187.

[3] X. Wang et al., "Sequential Hardware Trojan: Side-channel Aware Design and Placement," in *ICCD*, 2011, pp. 297–300.

[4] D. Agrawal et al., "Trojan Detection using IC Fingerprinting," in *IEEE Symposium on Security and Privacy (SP)*, 2007, pp. 296–310.

[5] C. Lavin et al., "RapidSmith: Do-It-Yourself CAD Tools for Xilinx FPGAs," in *FPL'11*, 2011, pp. 349–355.

[6] M. Banga and M. Hsiao, "VITAMIN: Voltage Inversion Technique to Ascertain Malicious Insertions in ICs," in *HOST*, 2009, pp. 104–107.

[7] J. Li and J. Lach, "At-Speed Delay Characterization for IC Authentication and Trojan Horse Detection," in *HOST*, 2008, pp. 8–14.

[8] M. Banga and M. Hsiao, "A Novel Sustained Vector Technique for the Detection of Hardware Trojans," in *VLSI Design*, 2009, pp. 327–332.

[9] F. Koushanfar and A. Mirhoseini, "A Unified Framework for Multimodal Submodular Integrated Circuits Trojan Detection," *IEEE Information Forensics and Security*, vol. 6, no. 1, pp. 162–174, 2011.

[10] G. T. Becker et al., "Implementing Hardware Trojans: Experiences from a Hardware Trojan Challenge," in *ICCD*, 2011, pp. 301–304.

[11] F. Benz et al., "BIL: A Tool-Chain for Bitstream Reverse-Engineering," in *FPL*, 2012, pp. 735–738.

[12] A. Moradi et al., "On the Vulnerability of FPGA Bitstream Encryption Against Power Analysis Attacks: Extracting Keys from Xilinx Virtex-II FPGAs," in *CCS*, 2011, pp. 111–124.

[13] ——, "Side-channel Attacks on the Bitstream Encryption Mechanism of Altera Stratix II: Facilitating Black-Box Analysis Using Software Reverse-engineering," in *FPGA*, 2013, pp. 91–100.

[14] Muehlberghuber et al., "Red Team vs. Blue Team Hardware Trojan Analysis: Detection of a Hardware Trojan on an Actual ASIC," in *HASP*, 2013, pp. 1–8.

[15] R. Karri et al., "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *IEEE Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[16] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[17] X. Wang et al., "Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions," in *HOST*, 2008, pp. 15–19.