Nonintrusive and Adaptive Monitoring for Locating Voltage Attacks in Virtualized FPGAs

Seyedeh Sharareh Mirzargar, Gaiëtan Renault, Andrea Guerrieri and Mirjana Stojilović
School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland
Email: mirjana.stojilovic@epfl.ch

In this work, we present an automated approach for locating power-wasting attackers in shared FPGAs. In this approach, the FPGA shell is responsible for, first, placing the user designs and, second, inserting voltage sensors in a nonintrusive and adaptive manner. We derive a metric for comparing transient voltage fluctuations recorded at different sensor locations and use it to locate the source of the highest disturbance. We implement and test our approach on a Xilinx Virtex-7 FPGA. In all our experiments, we successfully locate the attacker.

I. Introduction

Traditional FPGA usage model assumes one user controlling the entire FPGA, but with the current trend of enlarging FPGAs, this model will soon be replaced with a multi-tenant virtualized setting to avoid hardware underutilization [1]. In virtualized FPGAs, one part of the chip gets reserved for the *shell*, which handles resource management and sharing [1]; what is left is partitioned into regions (i.e., *roles*) to accommodate multiple users. Voltage attacks are the main obstacle in the way of deployment of shared FPGAs [2]–[8]. In these attacks, malicious users employ power-wasting circuits to lower the FPGA core supply voltage and cause timing faults in the colocated circuits or even reset the FPGA.

Previous research has shown that a network of on-chip voltage sensors can provide sufficient spatial resolution to locate the sources of voltage attacks [4], [9]. However, such a network requires considerable user resources and violates the desired isolation between the role space and the shell [10]. In this work, we present a new approach in which the voltage sensors are inserted after user designs and implemented with unused resources only. Additionally, we describe a novel metric for comparing sensor readings and locating the voltage attackers. Finally, we automate all the steps in a way that is compatible with the design flow employed by cloud service providers such as Amazon AWS [11].

II. Methodology

Commonly used voltage sensors consist of a ring oscillator (RO) and a frequency counter [4]. To keep the wire delay minimal and fit the RO inside a single CLB of the Xilinx 7-series FPGAs, we use eight-stage ROs, with only one inverting stage. And, we place the frequency counters inside the shell.

1) Adaptive placement: The first step of our automated sensor-insertion approach is entirely standard. We run Vivado to P&R the user design and export the design checkpoint (DCP). Then, we update the DCP by inserting the sensors. We are able to automate this step only thanks to the availability of RapidWright [12]: via its JavaAPI interface, we query all the CLBs in the user design and cluster them by their corresponding clock regions (CRs). Then, for each CR, we find the minimum bounding box (BB) covering the used CLBs and search for the first free CLB closest to the BB center. Once that CLB is found, we use RapidWright again to modify the DCP by inserting the sensor into it. Next, we feed the DCP back to Vivado, to route the sensor and connect it with the shell, perform design rule checks, and generate the partial bitstream.

2) Metric: As the power-delivery network impedance varies from one FPGA region to another [13] and considering the adaptive, and thus variable, placement of the sensors, we need a metric resilient to change in sensor location. The relationship between the frequency $f(x, y, t)$ of the sensor at location (x, y) and time instant t and the corresponding supply voltage $V(x, y, t)$, can be modeled as follows [4]:

$$f(x, y, t) = k(x, y) \cdot V(x, y, t) + f_O(x, y).$$

Since $k(x, y)$ and $f_O(x, y)$ are location dependent, it is not possible to reliably infer the relationship between supply voltages at two different locations by comparing the sensor frequencies. However, our experiments for quantifying the impact of process, voltage, and temperature (PVT) variations reveal that coefficient $k(x, y)$ is relatively constant, while the primary source of the spatial variability in sensor readings lies in coefficient $f_O(x, y)$. Computing the deviation of the sensor frequency from its average value:

$$\begin{aligned}\Delta\tilde{f}(x, y, t) &= f(x, y, t) - f_{\text{avg}}(x, y, t) = \\ &= k(x, y) \cdot \Delta\tilde{V}(x, y, t)\end{aligned}$$

and comparing it for two spatially-distant sensors s_i and s_j finally gives us an expression independent of $f_O(x, y)$:

$$\frac{\Delta\tilde{V}(x_i, y_i, t)}{\Delta\tilde{V}(x_j, y_j, t)} = \frac{\Delta\tilde{f}(x_i, y_i, t)}{\Delta\tilde{f}(x_j, y_j, t)} \cdot \frac{k(x_j, y_j)}{k(x_i, y_i)} \approx \frac{\Delta\tilde{f}(x_i, y_i, t)}{\Delta\tilde{f}(x_j, y_j, t)}.$$

Therefore, we use $\Delta\tilde{f}(x_i, y_i, t)$ as the metric for comparing the transient voltage fluctuations at different sensor locations.

To compute $\Delta\tilde{f}(x_i, y_i, t)$, we read sensors continuously and use sliding-window analysis. For every *transient* window (duration of about 10 µs) we report the worst-case (the lowest) recorded $f(x, y, t)$. For every *steady-state* window (duration of about 500 µs) we report the average sensor frequency.

III. Experimental Setup And Results

We use Vivado Design Suite ver. 2019.1, RapidWright ver. 2019.1, and Xilinx Virtex-7 FPGA VC707 Evaluation Kit. We do not perform the experiments on a cloud FPGA because we need access to the embedded temperature sensor to measure the impact of PVT variations on sensor readings; the average sensor frequency is 143.35 MHz. Similar to Yazdanshenas et al. [14], we reserve the two central clock regions for the shell and split the remaining space equally among four roles. The experiments are repeated ten times, each time with the sensor sampling period of 1.28 µs, and the transient window size of ten samples. The clock frequency is 200 MHz.

Five experimental setups are tested, each containing at least one power-wasting and one (or more) legitimate power-hungry circuits, as shown in Table I. The first power waster, *W1*, contains 70k asynchronous ROs [15], while the second, *W2*, contains 24k synchronous ROs [16]. With *W2*, we successfully run a timing-fault attack on a 128-bit ripple-carry adder. As power-hungry circuits, we use an image processing application, *H1*, and a synthetic noise generator, *H2*. Beside DSP slices and logic, *H1* uses the DDR memory and a soft-core processor. *H2* contains 1k 32-bit LFSRs [17].

We monitor the sensors during 655.36 µs (512 samples). Fig. 1 reports $\Delta\tilde{f}$ for setup e in Table I: Before activating the attacker, all $\Delta\tilde{f}$ are almost zero, suggesting no significant transient voltage fluctuations. During the attack, however, all $\Delta\tilde{f}$ drop by more than 60 MHz. The inset of Fig. 1 focuses on the transient sliding windows that cover the first 10 µs of the attack; it shows that, while all the sensors are affected by the transient voltage drop, those inside the CRs with the power-wasting circuit have the highest $\Delta\tilde{f}$. After the attack, the FPGA experiences a voltage overshoot as it takes time for the power-delivery network to lower the voltage in response to the deactivation of the attacker; hence, $\Delta\tilde{f}$ changes the polarity. After the recovery period, $\Delta\tilde{f}$ converges back to zero.

To locate the attacker in a setup with N occupied CRs and multiple roles, we first identify the N corresponding sensors. Then, we create a list $S = [s_0, s_1, .., s_i, .., s_{N-1}]$, $(s_i \neq s_j \, \forall (i \neq j) \in \{0, 1, .., N-1\})$, where s_0 has the lowest $\Delta\tilde{f}$ and s_{N-1} has the highest. Finally, we compute the user score, $Score_U$ as

$$Score_U = \frac{\sum_{s_i \in S} \Big((N-i) \cdot I_U \Big)}{\sum_{s_i \in S} (N-i)}. \quad (1)$$

Here, I_U is set to one for user U, if sensor s_i is inside their role region; otherwise, it is zero. In all the experimental runs, we find that the score of the power-wasting user is considerably higher than the score of the power-hungry user (2.21–7.40× higher), which shows, as expected, that the majority of the sensors with very low $\Delta\tilde{f}$ are inside the attacker.

IV. Conclusion

We describe an automated approach for locating the sources of voltage attacks in shared FPGAs. Our experiments on Xilinx Virtex-7 FPGA demonstrate the effectiveness of our approach.

TABLE I
Five experimental setups and their placement on the FPGA.

Setup	Circuits	Top left	Top right	Bottom left	Bottom right
a	*W1 & H1*	-	*H1*	*W1*	-
b	*W1 & H2*	*H2*	-	*W1*	-
c	*W2 & H1*	*W2*	*H1*	-	*W2*
d	*W2 & H2*	-	*W2*	*H2*	*W2*
e	*W2 & H1 & H2*	*W2*	*H1*	*H2*	*W2*

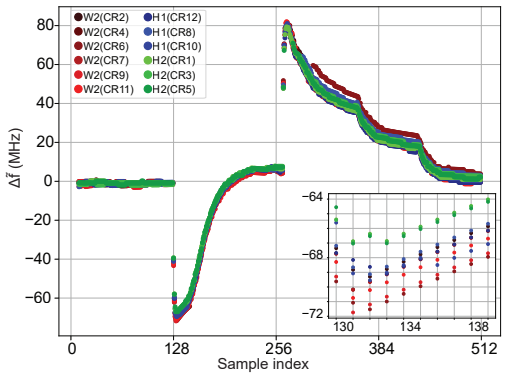

Fig. 1. The $\Delta\tilde{f}$ for setup e in Table I.

Future work will focus on porting and testing the methodology on a cloud FPGA, and adding the real-time decision making.

References

[1] M. Asiatici, N. George, K. Vipin, S. A. Fahmy, and P. Ienne, "Virtualized Execution Runtime for FPGA Accelerators in the Cloud," *IEEE Access*, pp. 1900–10, Feb. 2017.

[2] S. S. Mirzargar and M. Stojilović, "Physical side-channel attacks and covert communication on FPGAs: A survey," in *FPL*, 2019.

[3] T. M. La, K. Matas, N. Grunchevski, K. D. Pham, and D. Koch, "FPGADefender: Malicious self-oscillator scanning for Xilinx UltraScale+ FPGAs," *ACM TRETS*, pp. 15:1–15:31, May 2020.

[4] G. Provelengios, D. Holcomb, and R. Tessier, "Characterizing Power Distribution Attacks in Multi-User FPGA Environments," in *FPL*, 2019.

[5] D. Mahmoud and M. Stojilović, "Timing Violation Induced Faults in Multi-Tenant FPGAs," in *DATE*, 2019.

[6] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES," *TCHES*, pp. 44–68, Aug. 2018.

[7] O. Glamočanin, L. Coulon, F. Regazzoni, and M. Stojilović, "Are cloud FPGAs really vulnerable to power analysis attacks?" in *DATE*, 2020.

[8] D. G. Mahmoud, W. Hu, and M. Stojilović, "X-Attack: Remote activation of satisfiability don't-care hardware Trojans on shared FPGAs," in *FPL*, 2020.

[9] K. M. Zick and J. P. Hayes, "Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems," *ACM TRETS*, pp. 1–26, Mar. 2012.

[10] S. Trimberger and S. McNeil, "Security of FPGAs in data centers," in *IVSW*, 2017.

[11] *Amazon EC2 F1*, AWS, 2019. [Online]. Available: aws.amazon.com

[12] C. Lavin and A. Kaviani, "RapidWright: Enabling custom crafted implementations for FPGAs," in *FCCM*, 2018.

[13] R. O. Conn Jr, "Method and apparatus for measuring localized temperatures and voltages on integrated circuits," US Patent 5,795,068.

[14] S. Yazdanshenas and V. Betz, "The Costs of Confidentiality in Virtualized FPGAs," *IEEE T-VLSI*, pp. 2272–83, Oct. 2019.

[15] D. R. E. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on FPGAs using valid bitstreams," in *FPL*, 2017.

[16] T. Sugawara, K. Sakiyama, S. Nashimoto, D. Suzuki, and T. Nagatsuka, "Oscillator without a combinatorial loop and its threat to FPGA in data centre," *Electronics Letters*, pp. 640–42, May 2019.

[17] I. Giechaskiel, K. B. Rasmussen, and K. Eguro, "Leaky wires: Information leakage and covert communication between FPGA long wires," in *ASIACCS*, 2018.