Active Wire Fences for Multitenant FPGAs

Ognjen Glamočanin*, Anđela Kostić§, Staša Kostić§ and Mirjana Stojilović*
* School of Computer and Communication Sciences
§ Electrical and Electronics Engineering
EPFL, Lausanne, Switzerland
{ognjen.glamocanin, andela.kostic, stasa.kostic, mirjana.stojilovic}@epfl.ch

***Abstract*—When spatially shared among multiple tenants, field-programmable gate arrays (FPGAs) are vulnerable to remote power side-channel analysis attacks. Using carefully crafted on-chip voltage sensors, adversaries can extract secrets (e.g., encryption keys or the architectural parameters of neural network accelerators) from collocated tenants. A common countermeasure against power side-channel attacks is *hiding*; in hiding, the goal is to introduce noise and worsen the signal-to-noise ratio visible to the attacker. In a multitenant FPGA setting, hiding countermeasures can be implemented with an *active fence* placed between tenants. Previous work demonstrated the effectiveness of active fences built using NAND-based ROs. We enhance the state-of-the-art active fence implementation with novel *wire-based* power wasters, at no increase in resource overhead. Compared to an RO-based fence, our *active wire fence* makes the side-channel attack considerably more difficult. When using the RO fence to protect an AES-128 cryptographic module, we recovered all the bytes of the secret key with one million sensor traces, on average. In comparison, when using our novel wire fence, more than six million traces (an improvement of at least 6×) were required to recover all the bits of the secret key.**

***Index Terms*—FPGA, multitenancy, on-chip sensors, power side-channel attacks, protection**

I. Introduction

Field-programmable gate arrays (FPGAs) have a highly parallel, programmable hardware architecture that can implement and accelerate various applications. In recent years, the flexibility and energy efficiency of FPGAs has led to their integration into embedded devices, datacenters, and the public cloud. Today, many cloud-service providers (e.g., Amazon AWS, Microsoft Azure, Alibaba) offer FPGAs as remotely-accessible hardware acceleration instances [1]–[4]. Sharing of computational resources among multiple remote users, i.e., *multitenancy*, is a common approach for improving the efficiency of datacenter resource provisioning. With FPGAs in the cloud, considerable research efforts are put into extending multitenancy to FPGAs as well [5], [6].

Multitenancy on FPGAs presents a unique set of security challenges that cannot be fully addressed with solutions involving physical or logical isolation between tenants [7]. In spatially shared FPGAs, the tenants inevitably share the power distribution network (PDN), which enables a variety of electrical-level exploits: power side-channel attacks, denial-of-service attacks, fault-injection attacks, and covert communication [8]. In addition, the fine and low-level hardware control over the FPGA logic and wiring enables crafting malicious circuits (e.g., on-chip sensors for measuring shared supply voltage fluctuations [9]) that allow adversaries to execute attacks remotely. Recent work demonstrated remote power side-channel analysis (SCA) attacks on shared FPGA systems, including correlation power analysis (CPA) attacks on the Advanced Encryption Standard (AES) cryptographic modules [10]–[12], and power SCA attacks on neural network accelerators [13]–[17]. These attacks highlight the need for protecting FPGA users in a multitenant setting.

To protect against power SCA attacks on shared FPGAs, researchers proposed hiding techniques (i.e., reducing the signal-to-noise ratio), masking of victim operations, and detecting and preventing the deployment of voltage-sensing circuits [18], [19]. To implement hiding, Krautter et al. designed an *active fence* [20] consisting of ring oscillators (ROs), and placed it between the victim (in their case, an AES module) and the rest of the FPGA. The advantage of active fences is that they are independent of the victim's application and do not require modifications to the victim's design. Krautter et al. showed that the RO fence, of approximately the same size as the victim AES circuit, when activated with a pseudo-random number generator (PRNG), leads to a considerably higher attack effort: the number of traces needed to break a byte of the secret key using CPA increased by approx. 60×.

On FPGAs, noise can be generated in many ways. Here, we present a novel active fence design, offering better area efficiency and power side-channel security than the RO-based fence. We construct wire-based power wasters which are easy to replicate to build an *active wire fence* of arbitrary size. Similarly to previous work, we used a PRNG to activate the fence. To evaluate the effectiveness of the fence against a power side-channel attack, we collected millions of side-channel traces while an AES-128 cryptographic core was performing encryption. Our results show that, without any fence, the entire 128-bit key could be broken with approx. 30 thousand traces. With an RO fence, that number increased to one million traces, on average. Finally, with our novel active wire fence, more than six million traces (an improvement of 6×) were required to recover all the bits of the secret key. These results highlight the importance of developing alternative power wasters for building effective active fences.

This work is partially supported by the Swiss National Science Foundation (grant No. 182428).

II. Background and Related Work

A. *Power Side-Channel Attacks on Shared FPGAs*

In remote power SCA attacks on FPGAs, the two most commonly used on-chip voltage-fluctuation sensors are time-to-digital converters (TDCs) and frequency counters. Unlike hardened FPGA system monitors, these sensors measure voltage fluctuations indirectly, through logic delay changes. The primary component of a TDC is a tapped delay line, commonly implemented as a fine-delay chain of CARRY elements. In frequency counters, the basic element is a ring oscillator. As the signal propagation delay changes with the supply voltage, the readings of the TDC and RO-based sensors also change. Reading the sensor output at a constant rate allows for reconstructing power side-channel traces, the starting point of a power SCA attack. RO-based sensors require few resources but a relatively long time to make a single measurement [21]. In comparison, TDC sensors can detect voltage variations in intervals as short as a few nanoseconds [9]; hence, TDCs are more suitable for remote power SCA attacks where a high sampling rate is needed.

Zhao et al. successfully used RO sensors for a simple power analysis (SPA) attack against an RSA cryptomodule [22]. The same year, Schellenberg et al. demonstrated a successful CPA attack on an AES cryptographic module using traces from a TDC sensor [23]. Glamočanin et al. refined and ported the TDC sensor to an Amazon EC2 F1 cloud instance to showcase a successful CPA attack against an AES core [12]. Gravellier et al. used TDC sensors to execute a CPA attack against an AES bare-metal code running on the ARM CPU of an AMD Zynq-7000 SoC [11]. More recently, several research groups demonstrated successful reverse-engineering attacks against machine learning accelerators [13]–[17], showing that shared FPGAs are vulnerable to more than one type of power SCA attacks and, consequently, require proper mitigation techniques to enhance side-channel security.

B. *Power Wasters*

As another security vulnerability in shared FPGAs, researchers investigated the consequences of excessive power wasting. Gnad et al. [24] demonstrated that a large number of ROs activated in a repetitive pattern could reset the FPGA, resulting in a denial-of-service (DoS) attack. Mahmoud and Stojilović [25] showed that careful activation of ROs could introduce faults in neighboring circuits. Consequently, some cloud service providers, such as Amazon AWS, do not allow combinational loops in FPGA designs. As an alternative, researchers investigated power wasters free of combinational loops [26]–[28], typically less effective than ROs. In the active fence scenario, power wasters generate noise to reduce the signal-to-noise ratio (SNR); the noise must not be excessive, as the victim needs to operate correctly in its presence. If the fence is deployed by cloud service providers, both synchronous and combinational power wasters can be considered [20], and their number must be limited.

The most relevant power wasters to our work are the NAND-based ROs, used to build and evaluate an active fence in previous work [20]. The NAND-based ROs consist of a single look-up table (LUT), programmed as a NAND gate, where the output is connected back to one of the two inputs, while the free input acts as an enable signal. When the enable signal is active, the RO oscillates at a high frequency and draws current. An enhanced version of a NAND-based RO, ERO, was later proposed by La et al. and used to perform a remote DoS attack [28]. In the case of EROs, the output of one RO is connected to an unused LUT input of another nearby RO, thereby enhancing the effects of the switching activity thanks to the capacitance of the local interconnect. As we will later show, unlike EROs, our wire-based wasters primarily use longer, global wiring to enhance the effects of the RO switching.

C. *Active Fences*

Following the discovery of remote power SCA attacks in multitenant FPGAs, Krautter et al. [20] presented a new mitigation technique: active fencing. This hiding countermeasure employs ROs to generate noise in the PDN and, consequently, reduce the SNR measured by the attacker. The fence was designed as a set of RO banks (where each contains an equal number of ROs and can be independently enabled) placed between the victim and the attacker. The output signals of a PRNG or a TDC were used to control the fence activation. The result was a significant increase in the number of traces needed for a successful attack. Such a fence is easily portable and does not depend on the underlying victim design. However, the increased security comes at the cost of FPGA resources used by the fence. In previous work, the fence was dimensioned to occupy approximately the same number of slices as the victim (i.e., incurring 100% area overhead) [20]. We adopt a similar approach when comparing the efficiency of the wire fence to previous work: we implement a fence having the same type of ROs (NAND-based), use a PRNG to independently enable parts of the fence, and size the fence to occupy approximately the same number of resources as the victim to be protected.

III. Threat Model

Our work adopts the threat model commonly used in the literature on attacks in shared FPGAs [10], [12], [20], [22]. For security reasons, the users of the multitenant FPGA are physically and logically separated [29]. We assume an attacker attempting to impact the system's confidentiality by extracting secret information through the power side channel enabled by the shared PDN. In this threat model, the adversary can use a region of the FPGA to instantiate TDC sensors and measure the voltage changes caused by the victim, where the victim performs AES encryption with a secret key. The ciphertext is sent over a public channel accessible to the attacker, allowing them to execute a power analysis attack using the sensor traces and the ciphertext. Finally, the fence is implemented by the cloud service provider; therefore, the power wasters do not have to conform to security checks such as combinational loop detection.

IV. Active Wire Fence

In this section, we present our design of a wire-based power waster and the architecture of the active wire fence.

A. *Wire-Based Power Waster*

Our unit wire-based power waster consists of three components: a driver, a wire, and a sink. The driver acts as a *source*: it creates a high-frequency signal to drive the wire. What we refer to as the *wire* is, in fact, a connected sequence of FPGA routing resources (i.e., local and global wires, and routing multiplexers); when driven by the toggling signal, the wire consumes power and introduces noise. Finally, the *sink* is a termination point of the wire waster needed to prevent the FPGA compilation tool from optimizing it away.

To implement the source, we opt for a high-frequency signal generator: an RO implemented as a two-input NAND gate (occupying one LUT). Other pulse generators available in the literature, such as glitch generators [27], require considerably more resources than one LUT, thus negatively impacting the area overhead of the entire fence. When enabled to generate a high-frequency signal, the sources of the wire fence contribute to the overall noise [24]. In our wire waster, the sink is implemented as a buffer (one LUT). It, too, contributes to the overall power consumption of the wire waster, because its input is driven by a toggling signal. It can be noted that we do not attempt to maximize the contribution of the source or the sink, as we opt for the simple and resource-efficient implementations of the two terminal points of our wire waster. This approach facilitates the comparison with the RO-based fence and allows us to better estimate the impact of wires on the fence performance.

To ensure the use of global routing resources (i.e., wires spanning one or more FPGA slices), we place the sources and sinks sufficiently apart, and constrain them to the same FPGA column. After placing the source and sink of a power waster, we let the FPGA compilation tool complete the routing. To minimize the number of FPGA logic resources used by the fence, we do not impose any constraints on the signal routes (e.g., we do not force the routes to pass through LUTs or transparent latches at specific locations [27], [30]).

B. *Building a Wire Fence*

Before explaining the implementation of the wire fence, let us look at Fig. 1, illustrating a fence built solely from ring oscillators [20]. The ROs are organized in groups referred to as *banks*, which, in our implementation, have two configurable parameters: density (the number of ROs per FPGA slice) and size (the total number of slices occupied by the ROs). Hence, the total number of ROs in the fence equals the number of banks times the size and density. All ROs in the bank are controlled by the same enable signal, whereas each bank can be controlled independently from another. The fence is built by tiling multiple banks. As shown in Fig. 1, for the given area budget of the fence, we place the banks in column-major order, connecting one enable signal to each bank.

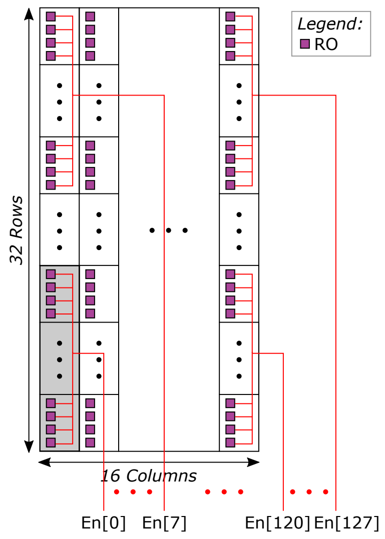

Fig. 1. RO fence occupying 32×16 FPGA slices; four LUTs are used per slice. Four vertically neighboring slices form a bank. In gray, bank with index zero. Each bank is controlled by a dedicated enable signal.

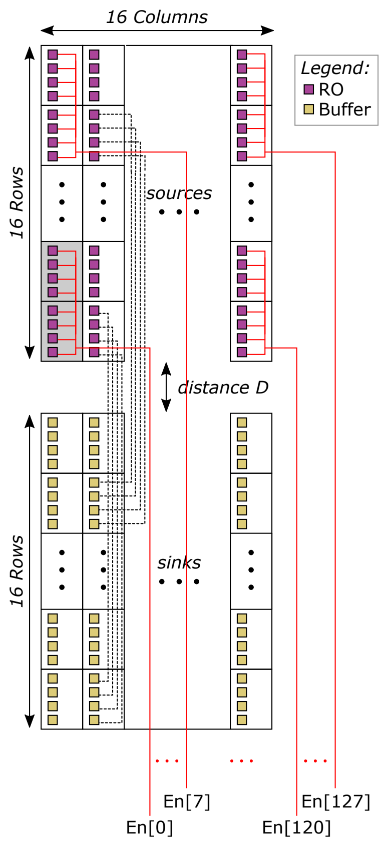

Fig. 2. Wire fence of 32×16 FPGA slices; four LUTs are used per slice. The top region is occupied by sources (ROs). The bottom region is reserved for sinks (buffers). The distance between the two regions is adjustable. Two vertically-neighboring slices form a bank. Each bank is controlled by a dedicated enable signal.

Our wire fence is shown in Fig. 2. The sources (ROs) are placed at the top; the sinks (buffers) are at the bottom. The distance parameter D corresponds to the number of unused slices between the *regions* occupied by the sources and the sinks. Similarly to the RO fence, the sources are organized in banks, characterized by their density and size, and placed in column-major order. The sources are enabled the same way as the blocks in the RO-fence (Fig. 1), with one enable signal per bank. To ensure the distance between a source and its corresponding sink (in terms of the number of slices vertically between them) is constant, we order the banks of sinks in the same way as their corresponding sources.

In the experimental evaluation (detailed in the next section), similar to Krautter et al. [20], we set the size of the fences to match the resource utilization of the design under protection. In our case, that value amounts to 2048 LUTs. Hence, the wire fence has, in total, 32 rows (16 for the sources and 16 for the sinks) and 16 columns of slices. Four LUTs are used per slice. Sources are grouped in 128 banks, each containing two slices (i.e., eight LUTs) and enabled independently. For a fair comparison, we implement the RO fence with the same resource utilization, resulting in twice as many ROs than in the wire fence. Fig. 1 illustrates the RO-fence implementation: 128 banks, each having four slices with four LUTs, organized in 32 rows and 16 columns.

V. Experimental Setup

For evaluating the fences, we choose Sakura-X [31], a board specially designed for power SCA evaluation and commonly used in research on attacks in both standalone and multitenant FPGAs [10], [15], [23]. It has two AMD FPGAs: Kintex-7 and Spartan-6. The former, also referred to as the target FPGA, houses the adversary and the victim. The latter, referred to as the control FPGA, aids in reducing unwanted noise by running the communication protocol between the target FPGA and the host machine. For the FPGA design compilation, we use AMD Vivado 2018.3.

Fig. 3 gives an overview of the system architecture, consisting of four main components. The open-source AES-128 running at 20 MHz is the victim design requiring protection [32]. The attacker's TDC sensor and FIFO, both running at 200 MHz, serve to record the side-channel traces. The attacker and the victim are physically and logically separated, according to the threat model. The fence is placed between the victim and the adversary. It is directly controlled with a 128-bit PRNG, implemented using a 7-bit Fibonacci linear-feedback shift register (LFSR) [33]. The LFSR generates a pseudorandom value from zero to 127, which is then decoded in hardware, and enables the corresponding number of fence banks. Finally, the controller is in charge of sending the plaintext and receiving the ciphertext, enabling or disabling the fence, calibrating the sensor, triggering the trace recording, and offloading the traces to the host machine.

The design floorplan is shown in Fig. 4. The attacker and the victim reside in separate FPGA regions. The sensor is placed at the edge of the attacker's region (top right), close to the victim, simulating a worst-case attack scenario. The active fence is located towards the edge of the victim's region (top

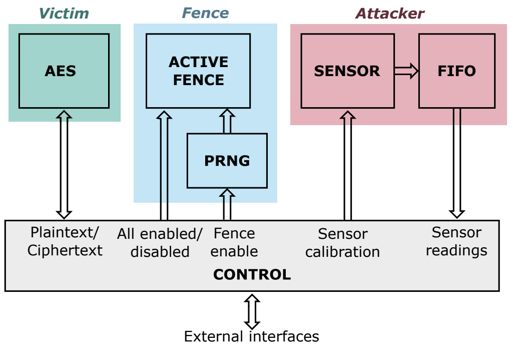

Fig. 3. System block diagram.

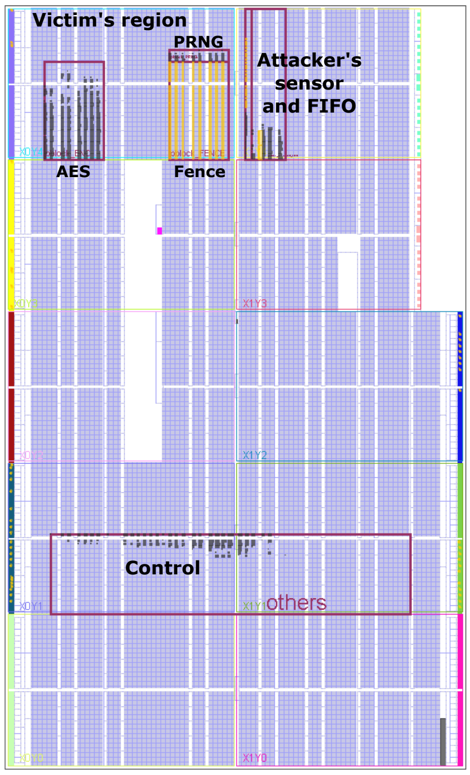

Fig. 4. Design floorplan, as seen from AMD Vivado.

left). The controller is placed away from the remaining logic, to minimize the noise.

To evaluate the efficiency of our active wire fence, we record sensor traces for three designs: (1) RO fence disabled, (2) RO fence enabled, and (3) wire fence enabled in place of the RO fence (using a different bitstream). For all three designs, we collect traces with a fixed key and chained plaintext: starting with an initial plaintext, we use the resulting ciphertext as the next plaintext. With the obtained traces, we run the CPA attack on the ninth AES round in steps and compute two metrics: the key rank metric of each byte of the 128-bit secret key [34], and the *key rank estimation* metric [35].

Using the key ranks obtained for all 16 bytes of the key, we compute the following additional metrics. First, we find the number of bytes broken when using all traces; this metric shows how successful an attack is in terms of broken bytes. Second, with all traces, we find the number of bytes for which the key rank is below or equal to 25, 50, and 100; this metric shows the key rank trends and how many bytes the attacker was close to or managed to break. Finally, we find the number of sensor traces for which the rank of (at least) one of the key bytes drops to zero and stays at zero for at least 10k traces (Metric-10k) and 30k traces (Metric-30k). These last two values will tell us how many traces are required to break at least one byte of the secret key.

However, evaluating the security of the individual key bytes does not necessarily reflect the security of the entire key. Therefore, especially when the key is not entirely broken, we use the key rank estimation metric and the CPA attack to estimate the remaining attack effort to break the entire key. For example, without side-channel information, the key rank estimation equals the entire key space, i.e., 2^{128} for AES-128. Alternatively, when the entire key is recovered, the key rank estimation drops to zero. In this paper, we use the histogram-convolution-based algorithm of Glowacz et al. [35] where the key rank is upper and lower bounded.

VI. Results and Discussion

A. Voltage Drop Comparison

In our first experiment, we compare the voltage drops caused by the following three types of power wasters: ROs, EROs, and the wire wasters described in Section IV. We build the RO fence from Fig. 1 (twice: once with ROs, once with EROs) and the wire fence from Fig. 2 (setting the distance parameter to zero). At the beginning of the trace recording, we disable the fence. Then, we enable all wasters in the RO (ERO, resp.) fence. In the case of the wire fence, we ran multiple experiments: with 25%, 50%, 75%, and 100% of the wasters enabled (by adjusting the number of active fence columns). Results, visualized in Fig. 5, show that, for the same amount of FPGA resources, EROs and wire wasters create significantly more pronounced voltage drops than ROs. In fact, with only 50% of the resources of the RO fence, the wire fence can create a voltage drop similar to that obtained with the fully utilized RO fence. Additionally, we observe that EROs and wire wasters create a similar voltage drop, even though the ERO fence has twice the number of ROs. From this, we can conclude that the long interconnects and the sinks used in our wire fence compensate for the smaller number of ROs.

B. Varying Distance D

To evaluate the impact of the distance D (between the source and sink regions in Fig. 2) on the voltage drop, we vary D from 0 to 150 and record the sensor output once the fence is enabled. In these experiments, the location of the sensor and the sources remains fixed. Fig. 6 shows the results, together with the RO-fence baseline from the previous experiment.

We can observe that as the distance D increases, the overall voltage drop does change, but not as much as one would expect. This result is rather unexpected because, if the distance is significantly larger, the wires connecting the sources and

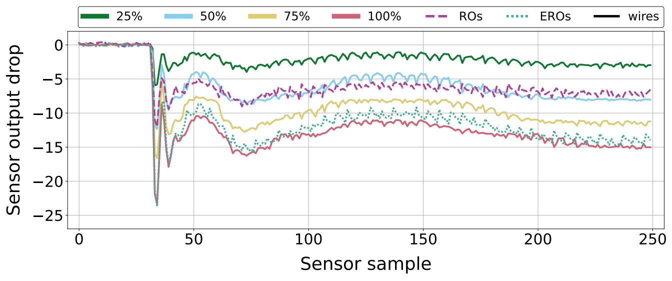

Fig. 5. Sensor readings drop (i.e., voltage drop) caused by the activation of RO/ERO wasters (dashed lines) compared to wire-based wasters (solid lines).

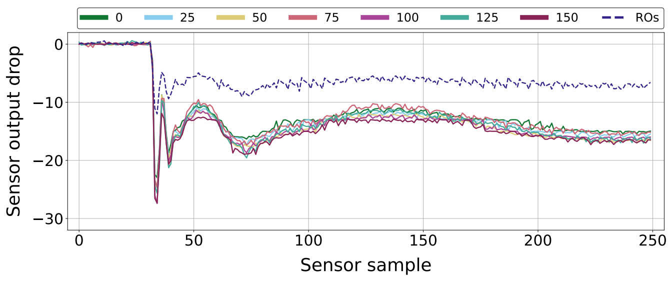

Fig. 6. Voltage drop for different distances D (in slices) with wire-based wasters (solid lines), compared to ROs (dashed line).

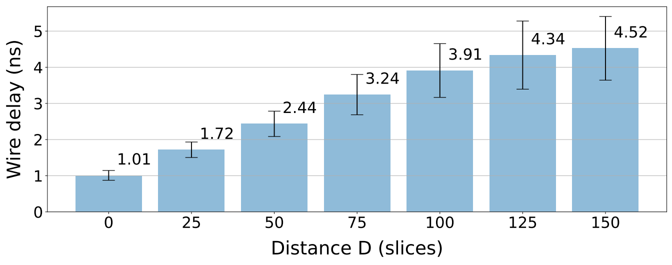

Fig. 7. Wire delay from source to sink of a wire-based waster extracted using Vivado 2018.3 static timing analysis, in the function of distance parameter D.

the sinks should intuitively be much longer. Thus, the voltage drop should be proportionally more pronounced. To investigate why this is not the case, we extract the routing delays from the fence sources to the sinks using Vivado static timing analysis, for each considered value of D. Fig. 7 shows the average delays and their standard deviations. We observe that the wire length, on average, increases proportionally with the parameter D. However, for the fixed location of the attacker sensor, higher D results in a greater distance between the sensor and the sinks, reducing the effect of the noise generated by a considerable part of the fence (the sinks and long wire segments close to them). Given the limited impact of increasing D on the voltage drop, and to minimize the area occupied by the fence, we set D to zero in the remaining experiments.

C. Power Side-Channel Attack Success

In our third and last set of experiments, we evaluate the effectiveness of the wire fence against a power side-channel attack in a multitenant FPGA setting. To that end, we collect power side-channel traces for the following scenarios: (1) RO fence disabled (three runs with 0.5 million traces), (2) RO fence controlled by the PRNG (three runs with 0.5 million and three runs with three million traces), and (3) wire fence controlled by the PRNG (three runs with 0.5 million, three runs with three million, and one run with eight million traces).

1) Three runs with half a million traces: The results of the CPA attack with 0.5M traces are summarized in Table I. The two central columns of the table show the number of traces needed to break one byte of the secret key. Metric-10k (Metric-30k, resp.) considers the byte broken when its key rank drops to zero and stays at zero for at least 10k traces (30k traces, resp.). The four right-most columns show the total number of key bytes for which the rank is below a certain threshold (100, 50, 25, and 0) at the end of the attack.

From Table I, we can observe that without any fence, it takes fewer than 10k traces to break the first byte of the key. Moreover, a CPA attack with 0.5 million traces can break all key bytes. With the RO fence, the number of traces needed to break the first byte according to Metric-10k increases by approximately 20×: 81–96k traces. According to Metric-30k, which requires the key rank to be stable for an even longer period of time, the number of traces to break a key byte increased even further (at least 100k). Between six and nine bytes were broken by the end of the attack. At the same time, all but one key byte had a rank lower than 100, showing that even though not all bytes were broken, their rank decreased considerably. Looking at the wire-fence results at the end of the attack, only four to nine key bytes had rank ≤100.

Next, we compute the *key rank estimation* metric [35]. The results for the three runs are aggregated and shown in Fig. 8. Because the key rank estimation is upper and lower bounded, we plot a shaded area indicating the entire range of the key rank estimation (min, max) observed across the runs. Additionally, the dashed and dotted lines correspond to the average lower and upper bounds across the runs. From Fig. 8, we see that, without a fence, the key is broken rather quickly: after approx. 30k traces. With the RO fence, the log key rank estimate drops to approx. 63, on average. With the wire fence, the key rank estimation remains very close to its maximum value. These results agree with those listed in Table I and show that the wire fence achieves higher power side-channel security than the RO fence when attacking with 0.5M traces.

2) Three runs with three million traces: Next, we repeat the experiments, this time to collect 3M traces. The results of the CPA attack are summarized in Table II. According to Metric-30k, to break the first key byte with the wire fence, approx. 2–10× more traces are required than with the RO fence. Moreover, with 3M traces, all key bytes are broken with the RO fence, while only seven to nine are broken with the wire fence.

Fig. 9 visualizes the log key rank estimation metric. Analyzing the results with the wire fence, we find that the key rank estimation dropped to 58 bits after 3M traces. With the RO fence, this level was reached after approx. 0.4M traces (7.5× faster). Lastly, with the RO fence, all key bytes were broken after approx. 1M traces, showing again that the wire fence provides superior power side-channel security.

TABLE I
CPA ATTACK RESULTS WITH 0.5M TRACES.

Setup	Run	Number of traces (10^3)		Bytes with key rank ≤			
		Metric-10k	Metric-30k	0	25	50	100
No fence	1	4	4	16	16	16	16
	2	3	3	16	16	16	16
	3	5	5	16	16	16	16
RO fence	1	89	115	9	14	15	15
	2	81	100	7	11	13	15
	3	96	190	6	15	15	15
Wire fence	1	N/A	N/A	0	0	1	9
	2	449	N/A	0	2	3	4
	3	N/A	N/A	0	0	1	5

TABLE II
CPA ATTACK RESULTS WITH 3M TRACES.

Setup	Run	Number of traces (10^3)		Bytes with key rank ≤			
		Metric-10k	Metric-30k	0	25	50	100
RO fence	4	115	136	16	16	16	16
	5	170	170	16	16	16	16
	6	37	37	16	16	16	16
Wire fence	4	759	864	8	14	16	16
	5	318	345	7	14	15	16
	6	148	394	9	13	14	14

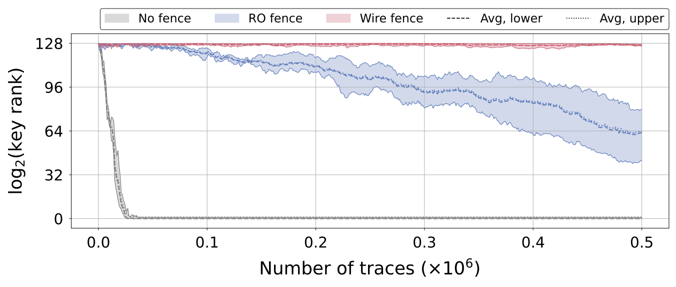

Fig. 8. Key rank estimation with 0.5M traces.

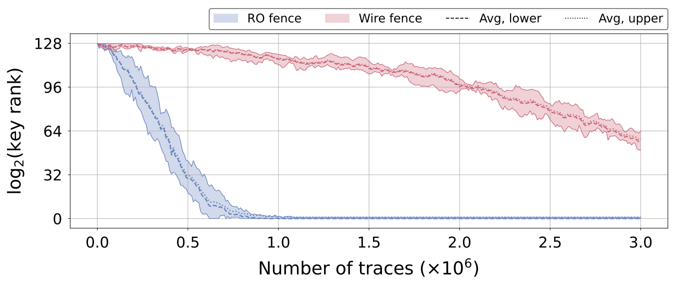

Fig. 9. Key rank estimation with 3M traces.

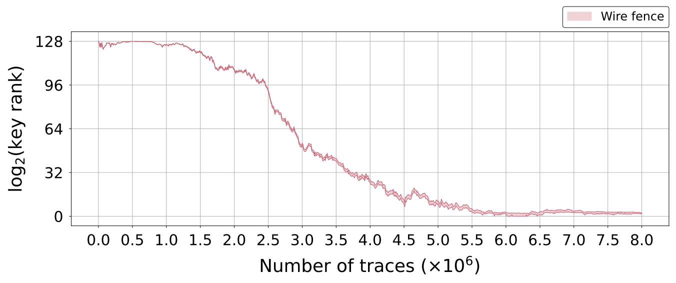

Fig. 10. Key rank estimation for the wire fence with 8M traces.

3) One run with eight million traces: In our last experiment, we collect 8M traces with the wire fence and plot the key rank estimation results in Fig. 10. The key rank drops to a value below three at approx. 6M traces and stays within the [0, 6) bound for the subsequent 2M traces. Even though not all 128 bits of the key are broken with 8M traces, the key rank estimation stays sufficiently low for an attacker to guess the remaining key bits with a limited effort. Therefore, we can conclude that with the wire fence, at least 6M traces were required to break the key: an improvement of at least 6× compared to the RO fence.

VII. Conclusions and Future Work

In this work, we presented a design of an active wire fence and demonstrated its ability to provide protection against remote power side-channel attacks in multitenant FPGAs. Our wire fence uses FPGA routing resources to draw more current and generate more noise than a fence built solely with ROs. Comparing the voltage drop resulting from the activation of the wire fence and the RO fence, we found that the RO fence compares to the wire fence of approximately half the logic resources. Therefore, when the space is limited, active wire fences are a better alternative to their RO counterparts. Comparing the side-channel attack effort to break a 128-bit AES key, we found that at least 6× more traces were required in a setup with a wire fence than with an RO fence. Future work will investigate combining enhanced ROs with the wire wasters, to further improve the efficiency and reduce the resource overhead of the active wire fence.

References

[1] *Amazon EC2 F1*, https://aws.amazon.com/ec2/instance-types/f1/, Amazon AWS, 2019, accessed: 2023-3-16.

[2] "Azure cloud services," https://learn.microsoft.com/en-us/azure/virtual-machines/sizes-field-programmable-gate-arrays, Microsoft, 2021, accessed: 2023-3-16.

[3] Alibaba, "Compute optimized instance families with FPGAs," alibabacloud.com/help/doc-detail/108504.htm, Alibaba, accessed: 2023-3-16.

[4] *Baidu Cloud*, https://cloud.baidu.com/product/fpga.html, Baidu FPGA, 2022, accessed: 2023-3-16.

[5] Y. Zha and J. Li, "Virtualizing FPGAs in the cloud," in *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, New York, NY, USA, Mar. 2020, pp. 845–58.

[6] S. Yazdanshenas and V. Betz, "The costs of confidentiality in virtualized FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, pp. 2272–83, Oct. 2019.

[7] T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. Levin, T. Nguyen, and C. Irvine, "Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems," in *Proceedings of the IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, May 2007, pp. 1–15.

[8] S. S. Mirzargar and M. Stojilović, "Physical side-channel attacks and covert communication on FPGAs: A survey," in *Proceedings of the 29th International Conference on Field-Programmable Logic and Applications*, Barcelona, Spain, Sep. 2019, pp. 202–10.

[9] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in FPGAs," in *Proceedings of the 21th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, USA, Feb. 2013, pp. 101–4.

[10] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Dresden, Germany, Mar. 2018, pp. 1111–16.

[11] J. Gravellier, J.-M. Dutertre, Y. Teglia, P. Loubet-Moundi, and F. Olivier, "Remote side-channel attacks on heterogeneous SoC," in *18th Smart Card Research and Advanced Applications Conference, CARDIS 2019*, Prague, Czech Republic, Nov. 2019, pp. 109–25.

[12] O. Glamočanin, L. Coulon, F. Regazzoni, and M. Stojilović, "Are cloud FPGAs really vulnerable to power analysis attacks?" in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Grenoble, France, Mar. 2020, pp. 1007–10.

[13] Y. Zhang, R. Yasaei, H. Chen, Z. Li, and M. A. A. Faruque, "Stealing neural network structure through remote FPGA side-channel analysis," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4377–88, Aug. 2021.

[14] A. S. Rakin, X. X. Yukui Luo, and D. Fan, "Deep-Dup: An adversarial weight duplication attack framework to crush deep neural network in multi-tenant FPGA," in *Usenix Security Symposium*, Aug. 2021, pp. 1919–36.

[15] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Power side-channel attacks on BNN accelerators in remote FPGAs," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 357–70, Apr. 2021.

[16] S. Tian, S. Moini, A. Wolnikowski, D. Holcomb, R. Tessier, and J. Szefer, "Remote power attacks on the versatile tensor accelerator in multi-tenant FPGAs," in *Proceedings of the 29th IEEE Symposium on Field-Programmable Custom Computing Machines*, Orlando, FL, USA, May 2021, pp. 242–46.

[17] V. Meyers, D. Gnad, and M. Tahoori, "Reverse engineering neural network folding with remote FPGA power analysis," in *Proceedings of the 30th IEEE Symposium on Field-Programmable Custom Computing Machines*, New York City, NY, USA, May 2022, pp. 1–10.

[18] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "Mitigating electrical-level attacks towards secure multi-tenant FPGAs in the cloud," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 12, no. 3, Aug. 2019.

[19] T. M. La, K. Matas, N. Grunchevski, K. D. Pham, and D. Koch, "FPGADefender: Malicious self-oscillator scanning for Xilinx UltraScale + FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 13, no. 3, p. 31, Sep. 2020.

[20] J. Krautter, D. R. E. Gnad, F. Schellenberg, A. Moradi, and M. B. Tahoori, "Active fences against voltage-based side channels in multi-tenant FPGAs," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Westminster, CO, USA, Nov. 2019, pp. 1–8.

[21] S. Moini, A. Deric, X. Li, G. Provelengios, W. Burleson, R. Tessier, and D. Holcomb, "Voltage sensor implementations for remote power attacks on FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, no. 1, pp. 1–21, Dec. 2022.

[22] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *Proceedings of IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, May 2018, pp. 805–20.

[23] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "Remote inter-chip power analysis side-channel attacks at board-level," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, New York, NY, USA, Nov. 2018, pp. 114:1–114:7.

[24] D. R. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on FPGAs using valid bitstreams," in *Proceedings of the 27th International Conference on Field-Programmable Logic and Applications*, Ghent, Belgium, Sep. 2017, pp. 1–7.

[25] D. Mahmoud and M. Stojilović, "Timing violation induced faults in multi-tenant FPGAs," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Florence, Italy, Mar. 2019, pp. 1745–50.

[26] G. Provelengios, D. Holcomb, and R. Tessier, "Power wasting circuits for cloud FPGA attacks," in *Proceedings of the 30th International Conference on Field-Programmable Logic and Applications*, Gothenburg, Sweden, Aug. 2020, pp. 231–35.

[27] K. Matas, T. M. La, K. D. Pham, and D. Koch, "Power-hammering through glitch amplification – attacks and mitigation," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Fayetteville, AR, USA, May 2020, pp. 65–69.

[28] T. La, K. Pham, J. Powell, and D. Koch, "Denial-of-service on FPGA-based cloud infrastructures—attack and defense," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, pp. 441–64, Jul. 2021.

[29] S. Trimberger and S. McNeil, "Security of FPGAs in data centers," in *2nd International Verification and Security Workshop (IVSW)*, Thessaloniki, Greece, Jul. 2017, pp. 117–22.

[30] A. L. Masle, G. C. T. Chow, and W. Luk, "Constant power reconfigurable computing," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, New Delhi, India, Jan. 2011, pp. 1–8.

[31] S. Lab, "Sakura-X side-channel evaluation board," https://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-X.html, 2021, accessed: 2021-9-20.

[32] *AES Encryption Core*, http://www.aoki.ecei.tohoku.ac.jp/crypto/, AIST and Tohoku University, 2019, accessed: 2023-3-16.

[33] "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators," https://docs.xilinx.com/v/u/en-US/xapp052, Xilinx, accessed: 2023-3-16.

[34] K. Papagiannopoulos, O. Glamočanin, M. Azouaoui, D. Ros, F. Regazzoni, and M. Stojilović, "The side-channel metrics cheat sheet," *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–38, Feb. 2023.

[35] G. Cezary, G. Vincent, P. Romain, S. Joachim, and S. François-Xavier, "Simpler and more efficient rank estimation for side-channel security assessment," in *International Workshop on Fast Software Encryption*, Istanbul, Turkey, Mar. 2015, pp. 117–29.