Shared FPGAs and the Holy Grail: Protections against Side-Channel and Fault Attacks

Ognjen Glamočanin*, Dina G. Mahmoud*, Francesco Regazzoni†‡ and Mirjana Stojilović*
*EPFL, School of Computer and Communication Sciences, Lausanne, Switzerland
†University of Amsterdam, Amsterdam, The Netherlands ‡ALaRI, Università della Svizzera italiana, Lugano, Switzerland

***Abstract*—In this paper, we survey recently proposed methods for protecting against side-channel and fault attacks in shared FPGAs. These methods are quite versatile, targeting FPGA compilation flow, real-time timing-fault detection, on-chip active fences, automated bitstream verification, etc. Despite their versatility, they are mostly designed to counteract a specific class of attacks. To understand how to address the problem of security in shared FPGAs in a comprehensive way, we discuss their individual strengths and weaknesses, in an attempt to identify research directions necessitating further investigation.**

***Index Terms*—FPGA, Multitenancy, Security**

I. Introduction

Given their ability to perform highly parallel computation with extremely flexible datapath and control, field-programmable gate arrays (FPGAs) are an ideal platform for accelerating many compute-intensive applications, e.g., artificial intelligence tasks, genomics, big data analytics, or image and video processing. As such, FPGAs have attracted commercial cloud service providers (CSPs), which now offer the latest generation FPGAs as remotely-accessible hardware acceleration platforms [1], [2].

To maximize the usage of FPGA resources, a number of multi-tenant virtualized FPGAs have been recently proposed [3]. Yet, they are *not* deployed, primarily due to the recently raised security concerns, supported by the researchers demonstrating denial-of-service, fault-injection, power side-channel, and crosstalk side-channel attacks on shared FPGAs.

The research community has tried to mitigate the problem, by making shared FPGAs and the cloud users less vulnerable to these attacks. However, the literature in the field is still quite fragmented and, consequently, the security issues of shared FPGAs are not yet addressed in a comprehensive way. With the goal of fostering future research in the field, in this work we review the existing protections against side-channel and fault attacks. We compare their main characteristics, we report their strengths and weaknesses, and discuss open problems.

II. Threat Model

A multi-tenant FPGA scenario assumes spatial sharing, where an FPGA can host multiple tenants simultaneously. Being intellectual property, the user code and data must be protected from other users that share the same FPGA; for instance, by enforcing logical separation between the tenants [4]. However, logical separation is not sufficient. On one hand, the tenants remain coupled through the device power distribution network (PDN). On the other hand, some of the data, e.g., the encrypted data, is commonly transmitted over publicly available communication channels, which can be observed by malicious parties. These security risks are real and, as recently demonstrated by researchers, exploitable by attackers.

Fig. 1 illustrates three types of electrical-level attacks in shared FPGAs. Some attacks aim to undermine the *confidentiality* of the system, by using the power side channel to obtain secret information from other users [5], [6]. Other attacks aim to break the *integrity* of the system by injecting faults in other users' computation [7], [8]. Finally, some attacks aim to threaten the *availability* of the remote system, by causing denial-of-service (DoS) [9]. In the following section, we discuss these three attack types in detail.

III. Attacks on Electrical Level

The low-level programmability of FPGAs allows attackers in multi-tenant scenarios to exploit low-level electrical phenomena for their attacks [10]. Two main categories of exploits exist: side-channel attacks (SCAs) and fault-injection attacks. Side-channel analysis is based on observing unintended leakage of information from a victim design. The malicious party can use a sensor to measure power variations on the chip (power side channel) or to deduce the value carried by a neighboring wire through the electromagnetic (EM) coupling effects (crosstalk side channel). A similar class of attacks are covert communication attacks; they use the same mechanisms, but require both a sender to send a message and a receiver with a sensor to read it. For fault-injection attacks, the adversary leverages power viruses to affect the PDN shared among tenants. Done aggressively enough, this can cause reset of the board (i.e., a DoS attack), and if done more precisely, this can cause a computational fault (i.e., a fault attack).

For a successful side-channel exploit, the most important component is the sensor. For remote attacks, ring oscillators (ROs), whose frequency of oscillation varies with voltage, are used for power SCAs [11] and for crosstalk SCAs [12]–[14]. They have also been leveraged for covert communication, where the sender is a CPU, GPU, or FPGA, and the receiver is an FPGA sharing the same power supply unit in a datacenter setting [15]. For a faster reaction time than ROs, delay-line sensors, similar to the one proposed by Zick et al. [16], are

This work is partially supported by the Swiss National Science Foundation (grant No. 182428) and the European Union Horizon 2020 research and innovation program under CPSoSAware project (grant No. 871738).

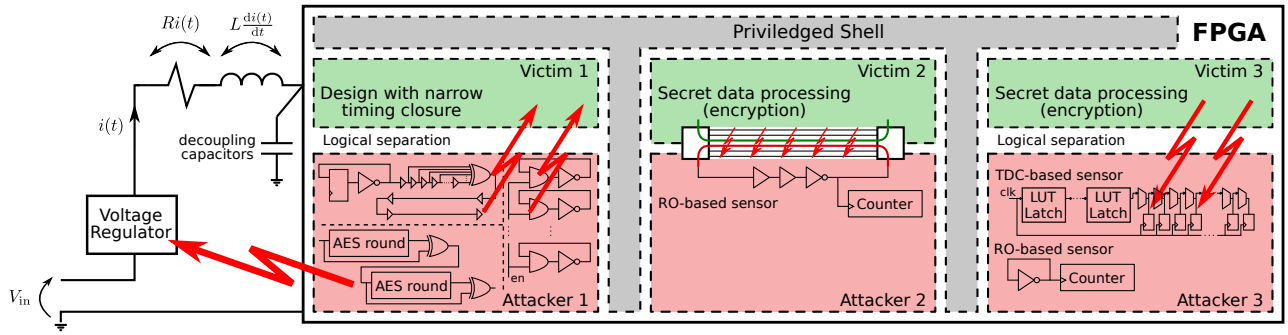

Fig. 1. Threat model for multi-tenant FPGAs. First, an attacker performing fault injection or denial-of-service attacks using various power hammering circuits. Second, an attacker exploiting the crosstalk coupling to perform a side-channel attack. Third, an attacker using voltage sensors for a power side-channel attack.

employed to sense fast voltage fluctuations [5], [17]. Delay-line sensors have even been used to demonstrate power SCAs on Amazon AWS F1 instances [6] and to recover the inputs to a neural network deployed on the same instances [18]. They have also been used to mount attacks against other integrated circuits on the same board [5] and against a CPU sharing the same system-on-chip [19].

Combinational ROs are not only used for sensing on-chip delay changes, but also as power viruses, which when used in a large enough grid and with specific activation patterns have been shown to cause board reset [9] and to inject faults [7], [8], [20]. Therefore, CSPs such as Amazon [1] have disallowed the use of combinational loops in designs deployed on their FPGAs. However, recent work has shown that other primitives, without the combinational loops, can be used for the same purposes. These include: sequential ROs, shift registers, dual-port RAM instances, glitch generators and even benign-looking circuits such as AES rounds [21], [22].

IV. Defence Strategies

A number of protections against one or more of the previously described attacks have been proposed. They employ very different approaches: from scanning the user bitstreams for virus signatures to modifying the look-up table design to render the FPGAs less sensitive to voltage variations. All of the proposed countermeasures have their strengths and weaknesses, which we will discuss in detail in Section V. Here, we introduce them and categorize them based on their underlying idea or the specific attacks that they target.

A. Bitstream Scanning

Similar to computer virus scanning, used to detect and prevent malicious software from running, scanning FPGA bitstream for malicious structures can help revealing structures of high risk to become a threat. For example, Beckhoff et al. have shown that the electrical short circuits can be detected by examining the partial FPGA configuration files [23].

More recently, Krautter et al. [24] and La et al. [25], followed this idea and developed bitstream scanners, which look for *signatures* of circuits that are of high risk to become security threats. La et al. [25] focus on circuits with high power-hammering capacity: cycles (combinational and sequential), high-fanout nets, glitch amplification designs, and short circuits, while Krautter et al. [24] also include the check for data-to-clock paths, present in delay-line voltage sensors. Both approaches take as input a (partial) bitstream file, extract the design netlist and build a corresponding graph, to search for malicious patterns in it. These bitstream scanners target only the Xilinx Ultrascale+ [25] and the Lattice iCE40 FPGA families [24], as they are among the few devices to have their bitstream reverse-engineering toolchains publicly available.

B. Voltage Transients Monitoring

Fast load transients, capable of causing significant voltage droops, are by far the largest contributors to FPGA circuit delay variation [9]. In processors, the techniques for handling load transients typically involve detecting core voltage droop, followed by the decrease in the working frequency [26]. FPGAs, however, rely on the PDN and its decoupling capacitors to respond to load transients, which can be insufficient [9].

Shen et al. investigated how to detect and mitigate the delay impact of load transients on FPGAs [27]. They designed a voltage-droop detector for an Intel Cyclone IV FPGA, which uses an adder as a delay-measurement unit (similar to carry-chain delay-line sensors in Xilinx FPGAs). The detector continuously measures the number of bits propagated and raises the droop signal if the number of bits propagated drops below a certain threshold. Setting this threshold requires calibration; its value may differ if the chip, the location of the droop detector, or the design frequency is changed.

Provelengios et al. [28] and Mirzargar et al. [29] investigated approaches suitable for cloud service providers: distributed sensing of on-chip voltage variations, previously studied by Zick et al. [16], for detecting and locating the highest voltage droops. On an Intel DE1 SoC, Provelengios et al. instantiated a network of evenly distributed RO-based voltage sensors and demonstrated that, in the 500 µs time period that contains the power-hammering attack, cubic interpolation algorithm can be used to reconstruct the voltage contours from the samples collected at the discrete sensor locations [28]. Mirzargar et al. focused on minimizing the intrusion of sensors into the user designs [29] and shortening the response time. They fully automated their approach, by augmenting the standard Xilinx FPGA design flow with an additional step—after the placement of the user design—that implements the RO-based

sensors using the remaining free resources only. With one sensor per clock region and the sensor readings in the period of 10 µs around the attack, they located the source of power-hammering activity on a Xilinx Virtex-7 FPGA.

C. Fault Detection

Increasing timing margins might be a solution against fault attacks. However, adding arbitrary timing margins does not ensure the absence of timing violations—it only reduces the risk. Therefore, additional countermeasures are needed.

Stott et al. [30] followed the approach of Razor [31] to build a timing-fault detector, consisting of a shadow register, comparator, and a fault latch. They also demonstrated that it is possible to automate the insertion of these detectors in an arbitrary FPGA application, using the information in the timing report and by modifying the design netlist.

Mahmoud et al. built a reconfigurable fault-detector circuit targeting a very specific attack scenario, in which the victim contains a *satisfiability don't care* hardware Trojan [20]. This Trojan, they showed, can be triggered in the presence of timing faults. Their fault detector consists of a pair of registers and a delay line, which is calibrated to a value close to the critical path delay of the victim. The detector controls an output multiplexer, which does not allow the victim to leak secret information under the attack (i.e., invalidates its output).

Some of the techniques to detect voltage droops, if calibrated carefully, can be used to detect timing faults as well. For example, Shen et al. leverage their voltage droop detector to build a clock edge suppressor, which, when needed, delays the next clock edge [27].

More advanced solutions have been proposed as well: Luo et al. [32] built a framework that controls the frequency of the target FPGA applications to avoid timing faults. The frequency scaling is implemented using an on-chip clock component (i.e., clock management tile of Xilinx Kintex-7 FPGAs), which can be configured statically or dynamically. In the static working mode, the framework takes a user-controlled frequency margin to compute and set the actual working frequency to a conservative value. In the dynamic working mode, an on-chip delay-line sensor is used to monitor the voltage, read the recommended operating frequency from a precomputed delay-frequency table stored in the on-chip memory, and reconfigure the clock management tile.

D. Power Side-Channel Leakage Reduction

Countermeasures against power SCAs have been extensively studied, but primarily in the context of the attackers having physical access to the device, and thus the possibility to connect the measuring equipment and observe current, voltage, or EM-field variations during the device's operation. These countermeasures fall into two main categories: *hiding* and *masking* [33]. In hiding, the focus is on reducing the signal-to-noise ratio captured by the measurements, e.g., by equalizing the data-related power consumption [34] or by increasing the noise component of the signal in the side channel. Masking requires processing algorithmically-randomized data, while maintaining the correctness of the circuit operation [35]. However, both suffer from considerable area overhead and vulnerability to higher-order attacks [33].

In the context of power SCAs on shared FPGAs, the works of Le Masle et al. [36] and Krautter et al. [37] are probably the most relevant. Both use a closed-loop control system to make the power consumption of the FPGA implementation less variable. Le Masle et al. use an on-chip RO-network to monitor the voltage [36]. Their control circuit is a proportional-integrative-derivative (PID) controller, whose PID constants are set so that the voltage measured by the sensors is kept approximately constant. As actuators, they use long routing interconnects, which consume power when their capacitance is charged and discharged. Krautter et al. create a side-channel protection by placing a *fence* composed of ROs between two neighboring FPGA tenants [37]. This active fence is controlled by a delay-line sensor. The authors chose the width of the fence to cover the victim circuit and its size to be equal to the size of the victim (an AES-128 module on a Lattice ECP5 FPGA).

E. Crosstalk Side-Channel Attack Mitigation

Two strategies for mitigating crosstalk SCAs have been suggested so far: first, by constraining the placement of the user logic and, second, by enhancing the FPGA routing algorithms to enforce free space around the important signals.

Huffmire et al. suggested using a spatial isolation mechanism called a *moat* and a controlled core-to-core communication mechanism called a *drawbridge* [38]. They first partition the design in nonoverlapping regions; the unused spaces between them become moats. Inside them, routing is disabled, except for the signals that use drawbridges to cross them. More recently, Yazdanshenas and Betz suggested wrapping the FPGA user applications (roles) with soft shells [39], in which the data that has to leave the role is encrypted. The cost of this, they show, is up to 80% higher latency compared to the unencrypted traffic and 20% less area available for the roles.

Luo et al. developed a hardware isolation framework named HILL [40], which ensures that the security-critical nets do not use long routing wires and isolates them from the remaining nets (by placing them in the center of the design and routing them before anything else). For the long wires that cannot fit within the design boundaries, the authors suggest adding a couple of unused adjacent long wires, to keep the potentially malicious nets away.

Seifoori et al. [41] worked on enhancing the PathFinder, a well-known FPGA routing algorithm used by the FPGA design tools [42], to ensure crosstalk-attack free designs by construction. Their approach requires the designers to label security-critical nets and to select one of the supported routing options, which differ in the number of neighboring long wires to leave unoccupied or in the choice of allowing or not the trusted nets to get near the security-critical ones.

F. Hardening FPGA Architecture and Implementation

Ahmed et al. suggested optimizing the look-up table (LUT) design, to render its input-to-output delays less variable with

the change of supply voltage [43]. They tried gate boosting the LUT, decoding the slowest two inputs of the LUT, and using separate voltage islands for the LUTs and routing. Although their work is not motivated by voltage attacks but dynamic voltage scaling, the idea of enhancing the FPGA architecture is certainly promising and worth exploring in the power side-channel attack context.

V. Evaluation and Discussion

To identify the advantages and drawbacks of the existing countermeasures, we look at how well they satisfy a number of relevant criteria. We summarize our findings in Table I and discuss them in the following subsections.

A. Generality and Interoperability

Two important characteristics of the countermeasures are *generality* (whether they are effective against more than a single class of attacks) and *interoperability* (whether, in principle, they can be combined with most of the other protections).

We find that no countermeasure is perfect. For example, the bitstream scanners considerably reduce the risk of the DoS and the fault attacks, but they fail to fully eliminate it—the proof of it being the increasing number of legitimate designs capable of power hammering [21], [44]. The crosstalk SCA countermeasures based on the separation between the tenants [38], [39] provide only partial protection, as crosstalk adversaries may be hidden, like Trojans, inside the designs themselves. The power SCA countermeasures help reducing the leakage, but do not eliminate it. The distributed voltage monitors help detecting voltage droops [28], [29], but may not be able to react sufficiently quickly during a DoS attack.

It is thus beneficial, when possible, to combine the protections. Combining countermeasures and the effects that a countermeasure against one type of attack can have on the resistance of a device against another type of attack is an open research topic [45]. However, even without performing a detailed analysis of the combinations of countermeasures, it is often possible to immediately state whether some countermeasures are mutually incompatible. For example, the bitstream scanners are incompatible with almost all the countermeasures that require RO sensors, delay-line sensors, or noise generators [20], [27], [28], [30], [32], [36], [37], because those circuits—unless hardened into the FPGA logic or placed by the CSPs—would match typical virus signatures and would thus be flagged as malicious. On the other hand, protections against the crosstalk attacks [40], [41], the FPGA architecture enhancements [43], and the distributed voltage monitoring by the CSPs [29] are the examples of those that do not have any immediate contraindication against combining them. Hence, we classify these countermeasures as, in principle, interoperable. It is, however, important to underline that further analysis is required to completely ensure that their combination does not alter the overall security of the device. In future work, we find it important to approach the countermeasure design in a holistic way and have them *offer a high degree of interoperability* with existing solutions.

B. Use of FPGA Resources

Not surprisingly, most countermeasures come at a cost (reducing the available area, the design frequency, or throughput). Exceptions are bitstream scanners, for obvious reasons. There are some reasonably inexpensive solutions: those detecting large changes in signal delays [20], [30] or those where the CSP uses the resources unoccupied by the tenants for voltage sensing [29], but they are a minority. In comparison, other approaches tend to be area- or energy-consuming, in particular when they require on-chip clocking and memory resources, large amount of logic for creating noise [36], [37], or the encryption and decryption of the incoming and outgoing data of every tenant [39]. Even though the datacenter FPGAs are rich in resources, we believe that *lightweight and energy-efficient countermeasures should be prioritized* in the future.

C. Passive and Active Protections

A natural way to classify the protections is to split them between *active* or *passive*, where active are those requiring real-time measurement of voltage or delay while the user design is operating. Interestingly, both types have attracted equal amount of research interest.

Active countermeasures may consume very few on-chip resources (i.e., timing-fault detectors) but also quite a lot of them (i.e., noise generators). Their main advantage, however, is the ability to monitor and thus react to the changing voltage and circuit delay. But, to do that correctly, they depend on a careful and long calibration process before the deployment. The calibration of this sort may not always be possible on a multi-tenant FPGA, as it is not guaranteed that the design will always be assigned the same cloud FPGA instance, nor it is guaranteed that the instance is free of other tenants while the calibration is taking place. The first issue could be addressed by fingerprinting the FPGAs [46]. For the second, adding an automated recalibration capability, potentially aided from the software side, to adjust to the changing conditions on the fly, could be interesting for future research. However, this would further increase the use of resources and require fast dynamic partial reconfiguration.

Since the characteristics of PDNs vary with the FPGA size, technology node, vendor, board design, etc., it should be important for the active countermeasures to show detailed *experimental verification and result characterization, using datacenter FPGAs and datacenter workloads*, for their efficacy and the applicability for the cloud to be thoroughly assessed. Unfortunately, almost none of the related publications provided such thorough experimentation. This type of evaluation is perhaps even more important for the active protections that involve noise generation, as increased noise may affect the correct operation of the FPGA co-tenants.

D. Who Should Deploy Them?

Many of the proposed protections are under full control of the FPGA users, because they are design-dependent and often require careful tuning for their successful operation. But there are alternative solutions, allowing the FPGA compilation tools

TABLE I
COMPARISON OF THE PROTECTION MEASURES.

Protections	Generality and interoperability					Use of FPGA resources			Type	Who should deploy them			Design disclosure	Portable
	crs	*pwr*	*dos*	*flt*	*iop*	*log*	*wire*	*clk*		*usr*	*vnd*	*csp*		
Krautter et al. [24]	✓	✓	◗	◗	◗	✗	✗	✗	Passive	✗	✗	✓	✗*	N/A
La et al. [25]	✓	◗	◗	◗	◗	✗	✗	✗	Passive	✗	✗	✓	✗*	N/A
Huffmire et al. [38]	◗	✗	✗	✗	✓	✓	✓	✗	Passive	✓	✗	✗	✗	✗
Yazdanshenas and Betz [39]	◗	✗	✗	✗	✓	✓	✓	✓	Passive	✗	◗	✓	✓	✓
Luo et al. [40]	✓	✗	✗	✗	✓	✗	✓	✗	Passive	✗	✓	✗	✗	✓
Seifoori et al. [41]	✓	✗	✗	✗	✓	✗	✓	✗	Passive	✗	✓	✗	✗	✓
Regazzoni et al. [35]	✗	◗	✗	✗	✓	✓	✓	✗	Passive	✓	✗	✗	✗	✓
Tiri et al. [34]	✗	◗	✗	✗	✓	✓	✓	✗	Passive	✓	✗	✗	✗	✓
Le Masle et al. [36]	✗	◗	✗	✗	◗	✓	✓	✓	Active	✓	✗	✗	✗	✗
Krautter et al. [37]	✗	◗	✗	✗	◗	✓	✓	✓	Active	✓	✗	◗	✗	✗
Shen et al. [27]	✗	✗	✗	✓	✓	✓	✓	✓	Active	✓	✗	✗	✗	✗
Provelengios et al. [28]	✗	✗	◗	✓	✓	✓	✓	✓	Active	✓	✗	✓	✓	✗
Mirzargar et al. [29]	✗	✗	◗	✓	✓	✓	✓	✓	Active	✗	✗	✓	✓	✓*
Stott et al. [30]	✗	✗	✗	✓	✓	✓	✓	✓	Active	✓	✗	✗	✗	✗
Mahmoud et al. [20]	✗	✗	✗	✓	✓	✓	✓	✗	Active	✓	✗	✗	✗	✗
Luo and Xu [32]	✗	✗	✗	✓	◗	✓	✓	✓	Active	✓	✗	✗	✗	✗
Ahmed et al. [43]	✗	◗	◗	◗	✓	✗	✗	✗	Passive	✗	✓	✗	✗	N/A

Legend:
crs) Crosstalk side-channel attack; *pwr)* Power side-channel attack; *dos)* Denial-of-service attack; *flt)* Fault attack; *iop)* Interoperability.
log) FPGA logic; *wire)* FPGA routing; *clk)* Clocking resources; *usr)* Users; *vnd)* FPGA vendors; *csp)* Cloud service providers.
✓ Yes; ◗ Partially; ✗ No; * Conditionally; N/A Not applicable.

or CSPs to contribute to design security. On one side, some crosstalk-attack defences rely on the FPGA design tools to produce protected designs. On the other side, bitstream scanners, tenant isolation, and distributed voltage monitoring require the support of the CSPs for their implementation. At first glance, having the protection put in place and maintained by the FPGA vendors or the CSPs could be the most convenient solution, which is why such protections should continue to receive research attention. Nonetheless, the lightweight and easy to implement strategies, to be deployed by the users themselves, will always remain valuable, especially when a high degree of user control is required by the target application.

E. Is Design Disclosure Required?

To reduce the risks of security issues, no CSP accepts the partial bitstreams from the users yet: the design rule checks and bitstream generation must be done on the servers of the cloud providers themselves [1], [2] so that at least some security verification can be performed (e.g., combinational loop detection). The inability to upload the bitstreams to the cloud, however, means that the design itself, at least to some extent, needs to be disclosed to the CSP. Bitstream scanners play an important role here, as they could be deployed together with an attestation system. For instance, if the bitstream is verified and attested by a third party, trusted by both the design owners and the CSPs, the design would not need to be disclosed (hence the conditional No in Table I).

In general, protections deployed by the users or FPGA vendors do not require design disclosure. Among those to be deployed by the CSPs, some would require access to the design files or, at least, the post-placement netlist for placing voltage sensors into the user regions [28], [29]. A solution could be to have a network of voltage sensors available as a hardened FPGA resource—something FPGA vendors could perhaps consider adding to future FPGA generations.

F. Portability

Another relevant characteristic of countermeasures is portability. If a protection does not require code rewriting or parameter tuning (calibration) with relatively minor design changes (e.g., the placement constraints or clock frequency) or when switching to an FPGA of a newer family, then such a technique has more chances to become a preferred choice. Protections such as masking or hiding, as well as those targeting vendor tools, satisfy our definition of portability. Additionally, the protections to be deployed by the CSPs could also be considered portable, at least from the user's point of view, as the CSPs still need to invest additional effort for providing the necessary infrastructure (hence the conditional Yes in Table I.)

VI. CONCLUSION AND FUTURE WORK

In this paper, we surveyed existing research on preventing or protecting against the power-side channel, crosstalk side-channel, and fault attacks in multi-tenant FPGAs. We categorized the attack countermeasures as passive or active (those requiring real-time voltage or delay measurements). We compared them using a number of criteria, such as generality

(the number of attack types they could be effective against), interoperability (the possibility of combining them with other protections, in principle at least), use of FPGA hardware resources, or the time and effort required to port them to other designs or devices.

We found that the majority of the countermeasures would not require the users to disclose their designs. We also found a number of relatively inexpensive countermeasures, at least in terms of use of FPGA resources, as well as a number of techniques that, at first glance, seem interoperable with other protections. Nevertheless, we have uncovered a number of issues that future research should try to address: low generality, in-depth exploration of interoperability and its implications, low number of fully automated solutions, and the general absence of the experiments in a real cloud setting.

Finding the right countermeasure—or a combination of them—remains an open problem. The best approach to solving it, we think, is through collaboration between and contribution from all the involved parties: researchers (by uncovering new attacks and developing new protection strategies), FPGA vendors (with enhanced FPGA architecture and tools), and the CSPs (by preventing or locating the attackers, while allowing everyone else to deploy their own protections, if they wish). We believe this paper to be an important step in this direction.

REFERENCES

[1] *Amazon EC2 F1*, Amazon AWS, 2019. [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/

[2] Alibaba, "Compute optimized instance families with FPGAs," Alibaba, alibabacloud.com/help/doc-detail/108504.htm.

[3] A. Vaishnav, K. D. Pham, and D. Koch, "A survey on FPGA virtualization," in *FPL*, 2018.

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

[5] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "Remote inter-chip power analysis side-channel attacks at board-level," in *ICCAD*, 2018.

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

[7] D. Mahmoud and M. Stojilović, "Timing violation induced faults in multi-tenant FPGAs," in *DATE*, 2019.

[8] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES," *IACR TCHES*, 2018.

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

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

[11] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *S & P*, 2018.

[12] C. Ramesh, S. B. Patil, S. N. Dhanuskodi, G. Provelengios, S. Pillement, D. Holcomb, and R. Tessier, "FPGA side channel attacks without physical access," in *FCCM*, 2018.

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

[14] I. Giechaskiel and J. Szefer, "Information leakage from FPGA routing and logic elements," in *ICCAD*, 2020.

[15] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "C3APSULe: Cross-FPGA covert-channel attacks through power supply unit leakage," in *S & P*, 2020.

[16] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in FPGAs," in *FPGA*, 2013.

[17] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," in *DATE*, 2018.

[18] S. Moini, S. Tian, J. Szefer, D. Holcomb, and R. Tessier, "Remote power side-channel attacks on CNN accelerators in FPGAs," 2020, arXiv: 2011.07603.

[19] J. Gravellier, J.-M. Dutertre, Y. Teglia, P. Loubet-Moundi, and F. Olivier, "Remote side-channel attacks on heterogeneous SoC," in *CARDIS*, 2019.

[20] 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.

[21] G. Provelengios, D. Holcomb, and R. Tessier, "Power wasting circuits for cloud FPGA attacks," in *FPL*, 2020.

[22] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, "RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions," in *FDTC*, 2019.

[23] C. Beckhoff, D. Koch, and J. Torresen, "Short-circuits on FPGAs caused by partial runtime reconfiguration," in *FPL*, 2010.

[24] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "Mitigating electrical-level attacks towards secure multi-tenant FPGAs in the cloud," *ACM TRETS*, 2019.

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

[26] M. S. Floyd, P. J. Restle, M. A. Sperling, P. Owczarczyk, E. J. Fluhr, J. Friedrich, P. Muench, T. Diemoz, P. Chuang, and C. Vezyrtzis, "Adaptive clocking in the POWER9™ processor for voltage droop protection," in *ISSCC*, 2017.

[27] L. L. Shen, I. Ahmed, and V. Betz, "Fast voltage transients on FPGAs: Impact and mitigation strategies," in *FCCM*, 2019.

[28] G. Provelengios, D. Holcomb, and R. Tessier, "Characterizing power distribution attacks in multi-user FPGA environments," in *FPL*, 2019.

[29] S. S. Mirzargar, G. Renault, A. Guerrieri, and M. Stojilović, "Nonintrusive and adaptive monitoring for locating voltage attacks in virtualized FPGAs," in *FPT*, 2020.

[30] E. Stott, J. M. Levine, P. Y. K. Cheung, and N. Kapre, "Timing fault detection in FPGA-based circuits," in *FCCM*, 2014.

[31] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *MICRO-36*, 2003.

[32] Y. Luo and X. Xu, "A quantitative defense framework against power attacks on multi-tenant FPGA," in *ICCAD*, 2020.

[33] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.

[34] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *DATE*, 2004.

[35] F. Regazzoni, W. Yi, and F.-X. Standaert, "FPGA Implementations of the AES Masked Against Power Analysis Attacks," in *COSADE*, 2011.

[36] A. L. Masle, G. C. T. Chow, and W. Luk, "Constant power reconfigurable computing," in *FPT*, 2011.

[37] 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 *ICCAD*, 2019.

[38] T. Huffmire, B. Brotherton, N. Callegari, J. Valamehr, J. W. R. Kastner, and T. Sherwood, "Designing secure systems on reconfigurable hardware," *ACM TODAES*, 2008.

[39] S. Yazdanshenas and V. Betz, "The costs of confidentiality in virtualized FPGAs," *IEEE TVLSI*, 2019.

[40] Y. Luo and X. Xu, "HILL: A hardware isolation framework against information leakage on multi-tenant fpga long-wires," in *FPT*, 2019.

[41] Z. Seifoori, S. S. Mirzargar, and M. Stojilović, "Closing leaks: Routing against crosstalk side-channel attacks," in *FPGA*, 2020.

[42] J. Rose, J. Luu, C. W. Yu, O. Densmore, J. Goeders, A. Somerville, K. B. Kent, P. Jamieson, and J. Anderson, "The VTR project: Architecture and CAD for FPGAs from Verilog to routing," in *FPGA*, 2012.

[43] I. Ahmed, L. L. Shen, and V. Betz, "Optimizing FPGA logic circuitry for variable voltage supplies," *IEEE TVLSI*, 2020.

[44] A. Boutros, M. Hall, N. Papernot, and V. Betz, "Neighbors from hell: Voltage attacks against deep learning accelerators on multi-tenant FPGAs," in *FPT*, 2020.

[45] F. Regazzoni, T. Eisenbarth, L. Breveglieri, P. Ienne, and I. Koren, "Can knowledge regarding the presence of countermeasures against fault attacks simplify power attacks on cryptographic devices?" in *DFT*, 2008.

[46] S. Tian, W. Xiong, I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "Fingerprinting cloud FPGA infrastructures," in *FPGA*, 2020.