A Visionary Look at the Security of Reconfigurable Cloud Computing

Mirjana Stojilović, *Senior Member, IEEE,* Kasper Rasmussen, *Senior Member, IEEE,* Francesco Regazzoni, *Member, IEEE,* Mehdi B. Tahoori, *Fellow, IEEE,* and Russell Tessier, *Senior Member, IEEE*

***Abstract*—Field-programmable gate arrays (FPGAs) have become critical components in many cloud computing platforms. These devices possess the fine-grained parallelism and specialization needed to accelerate applications ranging from machine learning to networking to signal processing, among many others. Unfortunately, fine-grained programmability also makes FPGAs a security risk. Here, we review the current scope of attacks on cloud FPGAs and their remediation. Many of the FPGA security limitations are enabled by the shared power distribution network in FPGA devices. The simultaneous sharing of FPGAs is a particular concern. Other attacks on the memory, host microprocessor, and input/output channels are also possible. After examining current attacks, we describe trends in cloud architecture and how they are likely to impact possible future attacks. FPGA integration into cloud hypervisors and system software will provide extensive compute opportunities, but invite new avenues of attack. We identify a series of system, software, and FPGA architectural changes that will facilitate improved security for cloud FPGAs and the overall systems in which they are located.**

***Index Terms*—Cloud computing, field-programmable gate array, secure computing, side channel attack, denial-of-service attack, fault injection.**

I. Introduction

Traditionally, cloud platforms have been based on a single type of computing devices: central processing units (CPUs). This homogeneity of hardware resources reflected itself in cost efficiency; buying thousands of very similar types of servers allowed cloud providers to reap the benefits of the economies of scale. The homogeneity of servers had other advantages as well: easy management and scheduling of resources, and simple development and deployment of applications and tools for debugging and tracing.

In recent years, however, cloud servers have gone through a significant change. They have progressively shifted to become heterogeneous platforms in which CPUs join forces with special purpose integrated circuits (e.g., Google's tensor processing units [1], graphics processing units (GPUs) and reprogrammable devices (i.e., field-programmable gate arrays (FPGAs)). One of the driving forces behind this change originated from the end of Moore's law coupled with the breakdown of Dennard scaling. While transistor size can still be reduced (though at a slower pace than before), with small transistor sizes and high operating frequencies, the power density increases significantly. Consequently, modern applications' continuously growing hunger for computing power can no longer be satisfied with general-purpose hardware only. We have entered an era in which computational performance growth will be fueled by specialized and heterogeneous hardware [1].

Besides energy efficiency, another important advantage of heterogeneous platforms is more consistent and predictable performance. Cloud applications previously were large monolithic services, but they have evolved into fine-grained, modular designs (microservices or serverless computing), where end-to-end tail latency is several orders of magnitude smaller (in the order of microseconds). Since missing latency requirements leads to cascading performance issues, datacenter hardware must be able to meet them. Traditional, general-purpose servers were not designed to accommodate such constraints. Heterogeneity, on the other hand, implies tailoring the hardware (and software) to the needs of datacenter applications, however strict.

Field-programmable gate arrays have emerged as the platform of choice for both regular and irregular forms of application parallelism. Their unique features are low-level hardware access, fine-grain programmability, and reconfiguration at runtime. Microsoft was among the first companies to recognize the potential of FPGAs for datacenter applications: Their Catapult servers relied on FPGAs to accelerate the Bing search engine [2]. Since then, a number of commercial cloud service providers (Amazon AWS [3], Alibaba [4], Baidu, Microsoft Azure [5], etc.) have started offering users remote access to datacenter FPGAs to develop and deploy their hardware accelerators.

Fine-grained control over the low-level FPGA hardware is, as it turns out, at the source of a number of electrical-level security issues. Today, we understand that FPGAs can empower a malicious user to execute a variety of remotely-controlled attacks: denial-of-service, fault injection, power side-channel, and crosstalk side-channel attacks. In this manuscript, in addition to describing the attacks in detail, we bring forward the multifaceted challenges of securely integrating FPGAs in the cloud, which are as relevant for FPGA vendors and developers as they are for cloud service providers and users. We discuss models for future cloud level use of FPGAs and elaborate

M. Stojilović is with the School of Computer and Communication Sciences, EPFL, 1015 Lausanne, Switzerland (e-mail: mirjana.stojilovic@epfl.ch).

K. Rasmussen is with the University of Oxford, United Kingdom (e-mail: kasper.rasmussen@cs.ox.ac.uk).

F. Regazzoni is with University of Amsterdam, Amsterdam, The Netherlands and Università della Svizzera Italiana, Lugano, Switzerland (e-mail: f.regazzoni@uva.nl; francesco.regazzoni@usi.ch).

M. B. Tahoori is with Karlsruhe Institute of Technology, Germany (e-mail: mehdi.tahoori@kit.edu).

R. Tessier is with the University of Massachusetts, Amherst, MA, USA 01003 (e-mail: tessier@umass.edu).

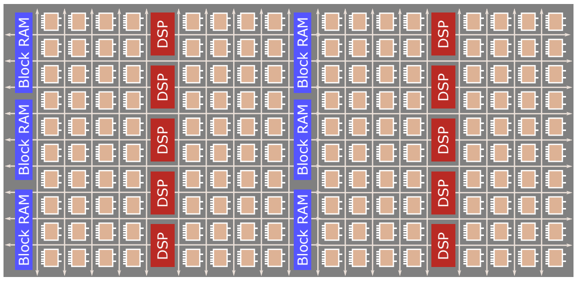


Fig. 1: An example FPGA architecture, showing a column-based layout of logic, memory, and DSP blocks.

on security techniques adapted for virtualized FPGAs in the cloud. Our visionary viewpoints provide insights into how FPGA-accelerated, heterogeneous cloud platforms will likely be used in the future.

The remainder of this manuscript is structured as follows. Section II describes the state-of-the-art in cloud FPGA use. An overview of cloud FPGA threats is described in Section III. Existing cloud FPGA attacks (Section IV) and remediations (Section V) are then described. In Section VI, trends in cloud FPGA systems are discussed followed by likely threats to these systems (Section VII). Suggested solutions to these challenges are described in Section VIII. We conclude the manuscript by summarizing lessons learned (Section IX) and offering closing thoughts (Section X). A list of the acronyms used in this paper is provided in the Appendix.

II. FPGAs As Compute Accelerators in the Cloud

In this background section, we introduce readers to FPGAs as compute accelerators in the cloud. We start by providing the basics of FPGA architecture (Section II-A). Then, we describe common heterogeneous cloud architecture (Section II-B) and the approaches cloud service providers use to expose the FPGA fabric to remote users (Section II-C).

A. Field Programmable Gate Arrays

Modern FPGAs consist of columns of logic blocks and heterogeneous hardened units such as block memories (BRAMs), digital signal processing blocks (DSPs), external memory interfaces, transceivers, phase-locked loops (PLLs) and even processor cores and GPU fabric [6].

Fig. 1 illustrates a small part of an FPGA die. The basic and most numerous building blocks of FPGAs are configurable logic blocks (CLBs), in AMD-Xilinx terminology, or logic array blocks (LABs), in Intel terminology. Each of these logic blocks is a cluster of look-up tables (LUTs), flip-flops, and carry propagation logic, which further facilitates the implementation of fast arithmetic circuits. The connectivity between FPGA building blocks is provided by numerous wires, which are grouped in horizontal and vertical routing channels, and routing switches, which can be configured to make connections between wires.

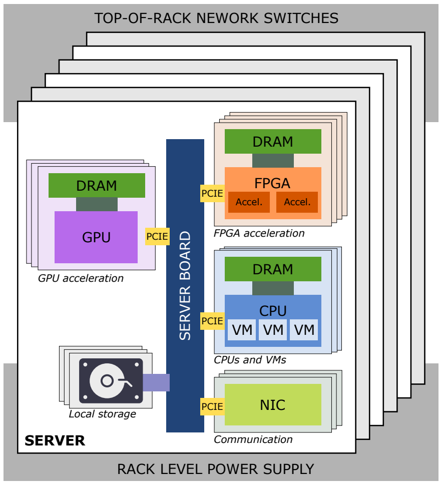


Fig. 2: A heterogeneous cloud server architecture.

B. Heterogeneous Cloud Server Architecture

The main computational building block of a datacenter is a *server*. The computational performance of today's servers is no longer driven by the growth of the number of CPU cores, but harvested from the heterogeneity of available computing units. In Fig. 2, we illustrate a heterogeneous server rack—a physical structure that holds tens of servers together, along with a rack-level power supply and top-of-rack (TOR) network switches. Thanks to TOR switches, servers can access the datacenter level network and reach any other server or the internet; additionally, the servers within one rack can communicate extremely efficiently with other servers in that same rack. A server itself can have a number of peripheral cards as well as local storage. A server configuration can be easily adapted by adding a suitable number of CPU, FPGA, GPU, or other accelerator cards via Peripheral Component Interconnect Express (PCIe) interfaces. Network interface controllers (NICs), which provide a networked connection between servers and TOR switches, are also integrated as PCIe peripherals. The configuration of servers across racks normally varies and is dictated by the targeted applications and desired performance.

The system illustrated in Fig. 2 corresponds to the *single node accelerator* model, deployed by a number of cloud providers [7] such as AWS, Huawei, Baidu, Tencent, Nimbix, and Alibaba. However, there are other ways in which FPGAs can be or already are deployed in datacenters. For instance, Microsoft uses FPGAs to intercept and accelerate network traffic to servers (a *bump-in-the-wire* configuration). In their Azure datacenters, multiple FPGAs in a rack can directly communicate with each other, allowing for datacenter-wide scalability of FPGA workloads [2]. These thousands of FPGAs are not only used to accelerate packet processing, but also Bing search and machine learning inference [8]. Baidu uses FPGAs to accelerate storage, SQL queries, data security, search engines, and artificial intelligence workloads.

A single node accelerator model is not the only way FPGAs could be exposed to users who would wish to deploy their custom accelerator in the cloud. An alternative is a *co-processor* model, in which an FPGA and a CPU co-reside on the same server card (e.g., Intel's Hardware Accelerator Research Program (HARP) platform, where the CPU and the FPGA are connected through a cache-coherent communication link) or even in the same package (e.g., as part of a system-on-chip (SoC) or a multiprocessor SoC (MPSoC)). The deployment of MPSoC server cards in commercial clouds will allow for improved performance for applications that do not require an extreme amount of computing resources. It will also allow an FPGA to offload most network configuration tasks and simplify FPGA orchestration with the local CPU.

C. FPGA Programming and Accelerator Deployment

Given their regular spatial architecture, FPGAs are perfectly tailored to implement highly parallel and deeply-pipelined circuits. Further, unlike on GPUs, hardware deployed on FPGAs can be of mixed granularity, ranging from single-bit (e.g., control lines) up to hundreds of bits (e.g., for AXI interfaces and memories). Designing an FPGA circuit implies fully describing the functionality of the desired hardware circuit. Most commonly, the process uses hardware description languages (HDLs), such as VHDL or Verilog, involving substantial work and competence. To bring FPGA programming closer to software developers, FPGA vendors have developed their own programming environments (e.g., supporting OpenCL [9]) and are also moving towards providing support for several languages and libraries for application spaces (e.g., Vitis from Xilinx and Intel's OneAPI Toolkit).

To decouple platform-specific FPGA hardware from user designs, cloud service providers employ a *shell-role* architecture [7], illustrated in Fig. 3. The *shell* often is comprised of a PCIe interface, a direct memory access (DMA) engine, a DRAM controller interface, virtual JTAG, and other health monitoring and image loading logic. The cloud service provider develops the shell design and specifies the interface to user-controlled regions (also called *roles*), in which custom accelerators can be quickly deployed using partial reconfiguration. For example, on AWS FPGA instances, the shell occupies approximately 20% of the FPGA resources. The separation between the shell and user regions allows for different privilege levels within an FPGA design and improves the reuse of user applications [7]. Additionally, the separation guarantees at least basic protection of the essential FPGA configuration and communication interfaces, which could otherwise be misconfigured or misused by customer applications.

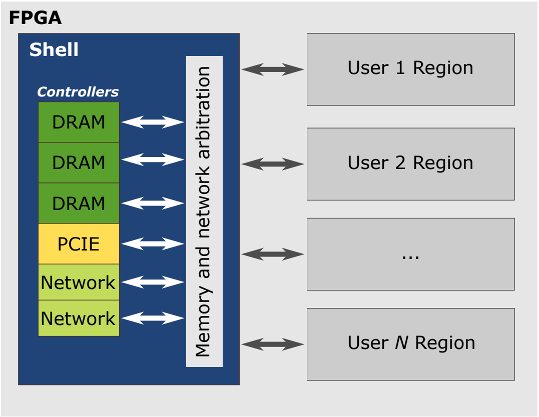


Fig. 3: Accelerator deployment model for cloud FPGAs, illustrating the separation between the shell and the user regions (i.e., user roles).

The shell generally presents either a host-centric or a shared-memory programming model. In both cases, the shell exposes a memory-mapped I/O (MMIO) control plane for software to manage the roles. The main difference between the host-centric and shared-memory models is whether the roles can issue their own direct memory accesses. The host-centric model, in which the user accelerators are unaware of the system memory map, is the more widespread. It yields simpler FPGA hardware, at the expense of increased communication latency between the CPU and the accelerator, in particular for applications that frequently perform pointer chasing (e.g., graph processing applications such as single source shortest path, etc.).

III. Threats to Cloud FPGAs

Since cloud FPGAs have been exposed to remote users, the attack surface on cloud infrastructures and cloud users has grown considerably. Some remote FPGA attacks aim to undermine the *confidentiality* of the cloud, e.g., by using side channels to extract secret information from other users. Others aim to break the *integrity* of the cloud by injecting faults into other users' applications. Last but not least, denial-of-service (DoS) FPGA attacks target the *availability* of cloud resources. These remote attacks, as we will soon see in detail, cover a wide landscape of security threats, including side-channel analysis attacks, intellectual property (IP) reverse engineering, hardware Trojan insertion and triggering, new covert channels, and accelerated device aging. In this manuscript, we will focus exclusively on successfully demonstrated remote FPGA attacks. Attacks which are impossible to demonstrate on a commercial cloud (e.g., because they require FPGA device tampering or substitution, modified software tools, untrusted or corrupted FPGA shell, etc.) or which are unrelated to the cloud FPGA use case (e.g., FPGA bitstream reverse engineering) are out of scope.

Remote FPGA attacks can broadly be categorized as electrical-level and system-level attacks. In the following subsections, we describe the two categories of attacks and their corresponding threat models.

A. Electrical-Level Attacks

Electrical-level attacks leverage the electrical coupling between the adversary and the victim. An effective way to achieve such coupling in datacenter FPGAs is via the shared power delivery network (PDN). Figs. 4 and 5 illustrate PDN sharing across a printed circuit board (PCB), FPGA package, and programmable logic. Due to the PDN's inductive, capacitive, and resistive components—included by design or as parasitics— it is almost impossible to completely eliminate PDN side-channel leakage. Furthermore, maintaining an exact

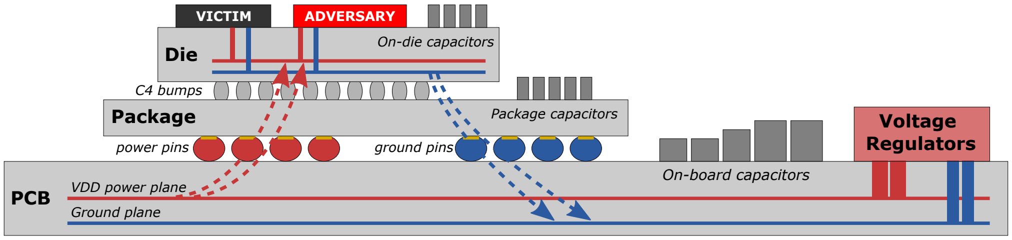


Fig. 4: High-level model of PDN sharing over a PCB (voltage regulators, power and ground planes, on-board decoupling capacitors), device package (power and ground pins, package capacitors), and FPGA programmable logic (C4 bumps, power and ground distribution grid). Dashed lines illustrate the flow of current, caused by the FPGA on-chip activity.

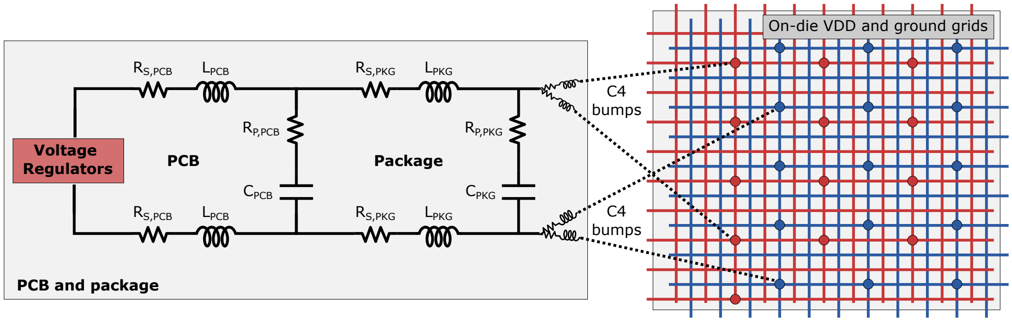


Fig. 5: Electrical model of the power delivery network illustrated in Fig. 4, emphasizing the serial and parallel parasitic resistance, capacitance, and inductance.

supply voltage level across the system, boards, chips, and individual transistors is generally impractical, if not impossible, irrespective of (data-dependent) transistor switching activities. PDN design usually maximizes reliability so that the amount of voltage drop is capped and limited to ensure proper chip timing at runtime. Several efforts have targeted reliable PDN design [10], [11], [12], [13].

The fine-grained hardware parallelism that makes FPGAs attractive for cloud applications, also poses security risks. The bit-level and wire-level hardware control provided by FPGAs gives malicious users the power to execute various electrical-level attacks [14]. For an electrical-level attack to be successful, an attacker must be able to either pick up the side-channel information generated by the target (i.e., the victim) or generate a signal (a disturbance) and inject it in the shared electrical medium, through which it propagates to the victim and causes either computational faults or the failure of the voltage regulator supplying the FPGA. The stronger the electrical coupling between the adversary and the victim, the higher the risk of a successful attack. For this reason, the most common threat model targeting PDN sharing assumes *FPGA multitenancy*, where multiple user applications run simultaneously on the same FPGA.

Fig. 6 summarizes the key features of the threat model of an electrical-level attack on a cloud FPGA. First, there is multitenancy. Second, the adversaries and the victims are physically and logically isolated (i.e., not reusing the same FPGA resources), although they share the same FPGA die and interfaces via the FPGA shell. Several attack variants are shown. In a fault-injection attack (scenario A in Fig. 6), the adversary uses specially designed FPGA circuits that draw high currents and, consequently, cause a significant disturbance in the shared supply voltage. Depending on the resulting voltage drop, the attack effects can range from injecting a fault in the victim's operation (and later exploiting it) to the reset or unresponsiveness of the entire cloud FPGA instance (a denial-of-service attack). In the power side-channel attack (scenario B in Fig. 6), the adversary deploys delay-sensing circuits to pick up the power side-channel leakage originating from the victims and uses it to extract their secrets (i.e., a cryptographic key, the images being classified by neural network accelerators, etc.).

On FPGAs, the shared PDN is not the only electrical side-channel medium: Programmable interconnects (i.e., the routing wires) have also been shown to leak information. The threat model of such an attack (scenario C in Fig. 6), commonly referred to as a *crosstalk* side-channel attack, considers that the victim is transmitting secret information over several cascaded long wires, spanning tens of columns or rows of FPGA logic elements; such long distances are not uncommon, especially for communication channels between the FPGA shell and the user partitions. Additionally, it is assumed that the attacker has access to the neighboring wires and continuously measures and analyzes their propagation delay, knowing that it correlates with the secret being transmitted [15].

Thermal effects bring about another class of electrical-level threats (scenario D in Fig. 6), which do not require spatial multitenancy. Unlike voltage-based attacks that typically last for a few microseconds (which is why they concern spatially collocated tenants), temperature changes take several orders of magnitude longer time to dissipate. Consequently, temperature can be used as a covert communication channel [16], [17]. The sender can enable a heater (e.g., a free-running oscillator

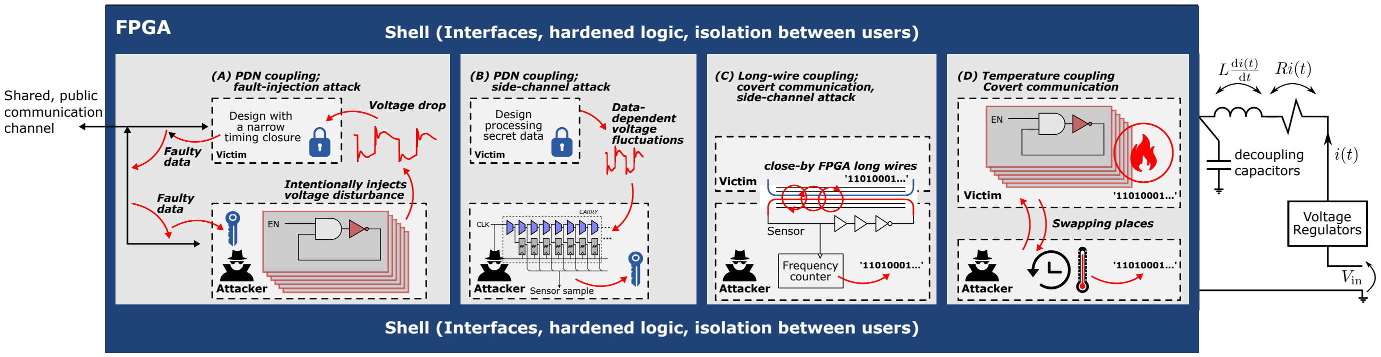

FPGA
Shell (Interfaces, hardened logic, isolation between users)
Shared, public communication channel
(A) PDN coupling; fault-injection attack
Design with a narrow timing closure
Victim
Voltage drop
Faulty data
Faulty data
Intentionally injects voltage disturbance
EN
Attacker
(B) PDN coupling; side-channel attack
Design processing secret data
Victim
Data-dependent voltage fluctuations
CLK
CARRY
FF
Attacker
Sensor sample
(C) Long-wire coupling; covert communication, side-channel attack
Victim
close-by FPGA long wires
'11010001...'
Sensor
Frequency counter
'11010001...'
Attacker
(D) Temperature coupling Covert communication
EN
Victim
'11010001...'
Swapping places
Attacker
'11010001...'
Shell (Interfaces, hardened logic, isolation between users)
$L\frac{\mathrm{d}i(t)}{\mathrm{d}t}$
$Ri(t)$
decoupling capacitors
$i(t)$
Voltage Regulators
V_{in}

Fig. 6: Threat model of an electrical-level attack on a cloud FPGA.

maximizing dynamic power) to raise the temperature, thus encoding and transmitting one bit of information; similarly, the sender can let the FPGA cool down to transmit the opposite value. As soon as the sender vacates the FPGA, the receiver may load their design with programmable sensors to read the on-chip temperature and infer the bit of information sent.

Bias temperature instability (BTI) effects are an example of an electrical-level phenomenon that leaves a trace for some time period. BTI effects physically deteriorate CMOS transistors, negatively impacting their switching speed [18]. They accumulate under voltage stress. Hence, FPGA resources holding a constant value (e.g., a secret key) for a long time are most affected. When the user vacates the FPGA and the voltage stress is removed, the transistors recover, slowly reverting to their previous faster state. An adversary residing on the same FPGA can monitor the BTI recovery process by measuring the propagation delay of the targeted FPGA resources over time [19]. Depending on how the propagation delay evolves over time, the adversary may infer the secret value previously imprinted by the BTI effects. Therefore, exploiting BTI effects is a form of a side-channel attack. Given that the attack targets specific FPGA resources, the threat model requires the adversary to have knowledge of the exact placement and routing of the victim design.

BTI effects on FPGAs can be significantly accelerated with thermal aging. Cook et al. [20] have shown that physically unclonable functions (PUFs) built from ring oscillators (ROs) are particularly sensitive to the frequency degradation caused by accelerated aging. By surrounding the FPGA ROs with short circuits (thus exposing them to extreme heat), their frequencies can be altered. Once the relative relationships of RO frequencies are adjusted, the overall PUF response can also be tuned. An adversary with the knowledge of the exact location of logic and wiring resources used by the RO PUF can use the targeted aging technique to imprint the desired PUF response, either for cloning the response of an authenticated device (an impersonation attack) or replacing it with an alternative of their choice. These results highlight the consequences of accelerated aging and warn against delay-based PUFs in cloud FPGAs.

B. System-Level Attacks

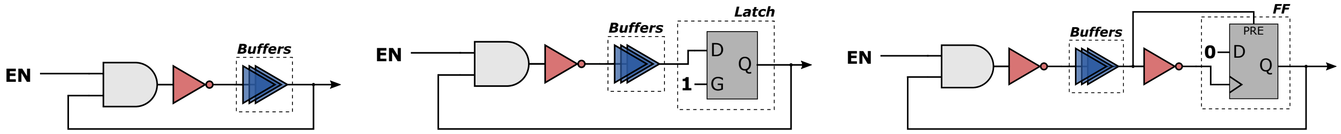
In the threat model of a system-level FPGA-assisted attack, an adversary uses the FPGA to attack other parts of the cloud-based system. For example, the FPGA may be used to corrupt portions of memory shared with a CPU or over-stress a communication bus shared with a CPU or other FPGAs. In these scenarios, an adversary requires access to the FPGA fabric and the ability to influence privileged CPU software (e.g., operating system, virtual memory manager, or device drivers). In the case that the adversary is able to control privileged FPGA logic, such as the shell, off-chip memory can be compromised (e.g., an adversary can intercept memory traffic via the shell). The host CPU, which is responsible for data transfer and for FPGA user region (role) configuration, may also be assumed to be untrusted and independent of any security mechanism provided by CPU trusted execution environments (TEEs). Hence, the vulnerability of the host CPU to TEE-targeted attacks is typically out of scope of FPGA-assisted system-level attacks. Recent work has focused on creating a TEE specifically for cloud FPGAs [21]. These architectures allow for the security of data generated by the FPGAs.

IV. Security Vulnerabilities of Cloud FPGAs

As introduced in the previous section, cloud FPGAs are susceptible to a variety of electrical-level attacks. To implement side-channel attacks, adversaries need one or more on-chip delay and voltage sensors. To inject a fault, force the FPGA to reset, or accelerate thermal aging, they need power-wasting circuits capable of drawing excessive current and causing a substantial drop in on-chip voltage. Given the low-level hardware control and bit-level programmability of FPGAs, many such circuits can be implemented. In this section, we introduce common FPGA malicious constructs (Section IV-A) before addressing cloud FPGA attacks in detail (Sections IV-B and IV-C).

A. Malicious FPGA Constructs

1) Voltage Sensors: FPGA circuits specifically designed to be highly sensitive to FPGA logic and routing delay variations are key enablers of remote power and crosstalk side-channel



(a) LUT-based RO

(b) LUT-Latch RO

(c) LUT-FF RO

Fig. 7: Various RO designs, suitable for on-chip voltage sensing. Buffers serve to control the RO frequency; they are optional.

attacks. Examples of such circuits are ring oscillators (ROs) and time-to-digital converters (TDCs).

An RO-based sensor is constructed by creating a loop whose frequency of oscillation is temperature and voltage dependent. An RO-sensor is typically enabled for some reference period (i.e., the measurement period), during which the corresponding number of pulses is counted. The measured frequency is then used to estimate the changes in voltage or temperature over time. When the temperature is approximately constant (as is the case for fast voltage transients), ROs sense local on-chip voltage variations. Such sensors have been used for power side-channel attacks (SCAs) [22], crosstalk SCAs [23], [24], and thermal covert communication [25]. They have also been leveraged to receive 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 [26].

Fig. 7 illustrates three ROs. The simplest design is comprised of an enable signal, a single inverting stage and, optionally, one or more buffers for adjusting the oscillation frequency. This sensor, being a combinational loop, can easily be detected; indeed, the AWS cloud service provider flags such designs and does not allow them to be implemented, precisely because of the security risks they pose. However, it is not difficult to build alternative RO-based sensor designs. In Figs. 7b and 7c we see two similar constructs, both free of combinational loops and thus more challenging to prohibit in a more general context.

ROs need long measurement periods for precision and are, therefore, unsuitable for side channels that rely on fast transients. Alternatively, time-to-digital converters (TDCs) are often used to overcome the limitations of ring oscillator-based sensors [27] and have been shown to effectively obtain side channel information on FPGAs [28]. In TDCs, each measurement reflects the delay of a circuit within a single clock cycle by observing how far through a tapped delay line a signal can travel during the cycle. This makes TDC sensors suitable for sensing short transient delay and voltage fluctuations on the order of a single clock cycle. For example, the TDC shown in Fig. 8 includes a chain of CARRY8 multiplexers used as delay stages. The adjustable delay blocks allow for delay path tuning. Delay-line sensors have even been used to demonstrate power SCAs on Amazon AWS F1 instances [29], to recover the inputs to a neural network deployed on the same instances [30], and to recover the information imprinted via BTI effects [19]. They have also been used to mount attacks against other integrated circuits on the same board [31] and against a CPU sharing the same system-on-chip [22].

2) Power Wasters: Ring oscillators are not only suitable for voltage sensing but also for drawing power. They are particularly efficient power viruses (i.e., power wasters), thanks to their high oscillation frequency and small footprint, as well as the ease of instantiating many of them. Similarly to RO-based sensors, several variants of RO-based power wasters can be built (Figs. 9a, 9b, and 9c). One would be tempted to think that detecting and preventing combinational loops would be a solution; however, the problem is significantly more difficult: Power wasters can be built in a variety of ways, and even benign-looking circuits can draw excessive current. In Figs. 9d, 9e, 9f, and 9g, we see examples of different power wasters—circuits harnessing glitches, CARRY8 blocks, shift registers, and even Advanced Encryption Standard (AES) encryption rounds [32], respectively. They have been successfully used for remote fault injection or denial-of-service attacks. For even more aggressive attacks (e.g., targeted RO PUF aging in impersonation attacks), short circuits are required [20].

B. Cloud FPGA Attacks

1) Fault Injection and Denial-of-Service: The use of on-FPGA power wasters to induce timing delay faults in victim circuits has been extensively studied. Characterizations of power waster voltage effects have been performed for localized attacks [48], dynamic and transient attacks [49], and attacks targeting the entire FPGA [36]. These characterizations show that voltage manipulations are possible chip-wide with an isolated power waster due to the FPGA's shared PDN. Recently, Alam et al. [44] showed that allowing a user to intentionally cause write collisions in FPGA dual-port block RAMs can also induce voltage and temperature fluctuations and result in circuit faults. Faults in victim circuits have also been induced using power wasters based on AES [50], [40], and glitch generators based on XOR gates [34] [45].

An assortment of victim circuits have been targeted for fault injection. Krautter et al. [39] examined the possibility of injecting faults into an AES core at a number of operating frequencies and circuit minimum slack values. In Mahmoud and Stojilović [37], a fault-inducing attack on true random number generators (TRNGs) using ROs was described. The ROs were placed adjacent to TRNGs and TDCs were used to evaluate induced delay changes. In Provelengios et al. [42], Rivest–Shamir–Adleman (RSA) encryption was successfully attacked by enabling power wasters and inducing timing faults. The faulty output was analyzed to determine the secret RSA key [42]. Several attempts have been made to inject faults into machine learning circuits to cause mischaracterization

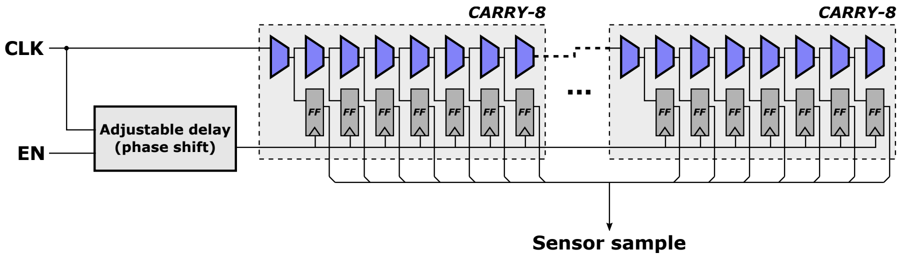
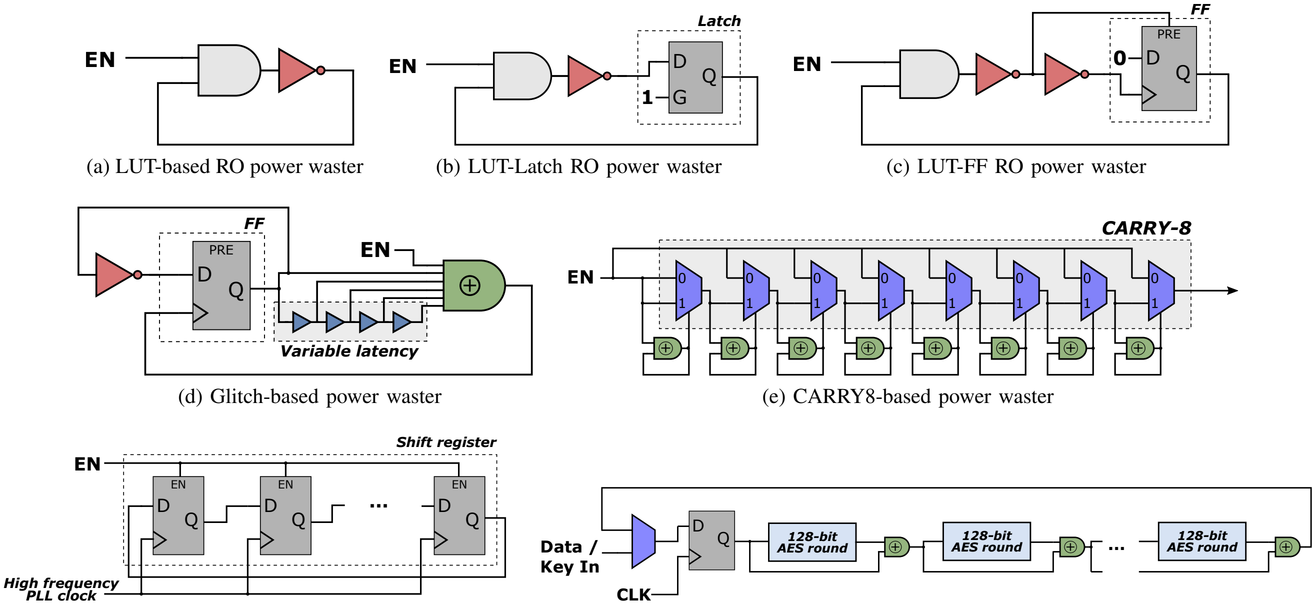


Fig. 8: FPGA implementation of a time-to-digital converter (TDC), with carry propagation logic serving as a delay line.



(a) LUT-based RO power waster

(b) LUT-Latch RO power waster

(c) LUT-FF RO power waster

(d) Glitch-based power waster

(e) CARRY8-based power waster

(f) Shift register-based power waster, initialized with alternating zeros and ones

(g) AES-based power waster

Fig. 9: FPGA power wasting circuits which, when instantiated in large numbers, can cause fault injection or FPGA reset.

[45], [47], [46]. Fault injection via voltage manipulation in machine learning is challenging due to model redundancy and the significant timing margins employed by FPGA physical design tools [45]. Other application attacks include stealthy FPGA Trojan triggering [41], and FPGA-to-CPU undervolting for injecting faults in CPU code execution [38].

If a sufficient supply of power wasters are simultaneously enabled, the regulators supplying power to the FPGA will be reset. Gnad et al. [33] showed that the sudden activation of thousands of ROs can drive Xilinx FPGAs into reset, requiring a bitstream reload. Although this attack results in a denial of service, it is not capable of stealthily extracting information from an unsuspecting circuit. Provelengios et al. [50] demonstrated that board failure for an Intel Stratix 10 FPGA can occur in as little as 20 microseconds, necessitating an effective remediation approach.

Table I summarizes the research on PDN fault attacks. It compares the attack objectives, attack types (intra-FPGA or intra-SoC), victim applications, malicious designs deployed, and the FPGA platforms (including the public cloud) on which the attacks were demonstrated.

2) Side-Channel Attacks: In addition to fault injection, on-chip voltage fluctuation caused by victim circuit activity can be monitored to extract information. A variety of voltage fluctuation sensors have been crafted and demonstrated to work (as shown in Figs. 7 and 8) [27], [28], [36], [56], [57]. For example, Zhao and Suh [22] demonstrated that RSA encryption activity on a microprocessor could be detected using ROs in the FPGA fabric when both devices share the same power source. More common attacks on encryption occur when both the victim circuit and sensors are located in the FPGA fabric. AES key information was extracted on a stand-alone FPGA board [28] and AWS EC2 F1 [29] using a TDC. A TDC was used to extract a black-and-white image input to a binary neural network (BNN) circuit [30], where tiny voltage fluctuations were used to differentiate between black and white pixels. In Tian et al. [54] a TDC is used to identify operational phases and parameters of a versatile tensor accelerator.

SCAs that use voltage and electromagnetic effects have been demonstrated using one or more FPGAs. In Gnad et al. [58], ring oscillators are periodically enabled to reduce on-FPGA voltage. A collection of TDCs is used to measure small voltage changes as a changed logic value. A similar approach was used to communicate information across multiple dies (super logic regions) in an FPGA [59]; in this work, multiple ROs are used

TABLE I: Comparison of PDN fault attacks. In bold, attacks demonstrated on a public cloud.

Attack objective	Type	Target circuit (the victim)	Malicious circuits (the attacker)	Evaluation platform
Denial of service	Intra FPGA	Host FPGA [33], [34], [35]	Single-stage LUT-based RO [33] Glitch generator and long wires [34] ROs with transparent latch [35] ROs with FFs [35] ROs through carry chain logic [35] Glitch amplification [35]	Virtex 6 (ML605) [33] Kintex 7 (KC705) [33] Zynq 7020 (Zedboard) [33] Zynq UltraScale+ (Ultra96) [34] **Virtex UltraScale+ (Amazon AWS)** [35] Virtex UltraScale+ (Alveo U200) [35]
Fault injection	Intra FPGA	Adder [36], [32] RNG [37]	19-stage ROs as voltage sensors [36] Single-stage ROs [36], [37] ROs with FFs [32] Shift registers[32]	Cyclone V (Terasic DE1-SoC) [36], [32] Aria 10 GX (Terasic DE5a-Net) [32] Virtex-7 (VC707) [37]
Fault injection	Intra SoC	Software routines: multiplication [38] AES [38]	Single-stage ROs [38]	Zynq UltraScale+ (Genesys-ZU) [38]
Recover the key	Intra FPGA	DFA on AES [39], [40] HW Trojan infected AES [41] Adder [42] RSA [42] DFIA on AES [43]	Single-stage LUT-based RO [39], [41], [42], [40], [43] 19-stage ROs as voltage sensors [42] AES [40] ISCAS'89 s1238 benchmark [40]	Cyclone V SoC (Terasic DE1-SoC [39], [41], [42], [40], Terasic DE0-Nano-SoC [39]) Aria 10 GX (Terasic DE5a-Net) [42] Lattice Semiconductor iCE40HX8K [40] Stratix 10 SX SoC (DE10-Pro) [40] Spartan-7 (Arty S7) [43]
Degrade network inference accuracy	Intra FPGA	CNN [44] MobileNet-V1 [45] DNN LeNet-5 [46] ResNet-20 [47] VGG-11 [47], MobileNetV2 [47]	Dual-port RAM memory collisions [44] TDC for timing the attack [46], [47] Single-stage ROs [45] ROs with transparent latches [46], [47] Clock-gated garbled XORs [45] Clock-gated hybrid toggling logic [45]	Artix-7 (Nexys 4 DDR) [44] Stratix 10 (Terasic DE10-Pro) [45] Pynq-Z1 [46] Zynq UltraScale+ (ZCU104) [47]

to detect communicated values. Finally, when a power supply is shared, it was shown that communication via on-FPGA voltage manipulation can be made across FPGA chips [26] and even across boards that contain FPGAs [26]. It was found that cross-device communication is more effective when the on-chip voltage of the receiver is stressed using RO power wasters.

Side-channel attacks in FPGAs can also be carried out using adjacent long FPGA wires. It has previously been shown [60], [23], [24] that the delay of a wire differs slightly if the adjacent wire carries a logic ‘0’ or a logic ‘1’. This difference can be exploited to extract an AES encryption key [23] from an unsuspecting victim.

Finally, Drewes et al. [19] analyzed the side-channel created by BTI effects. On an AWS EC2 F1 instance, they deployed TDC sensors to track the recovery behavior of the FPGA routing wires and multiplexers previously exposed to accelerated BTI effects. They observed a difference in the recovery behavior, which correlates with the type of BTI effect (positive or negative) to which the wires and multiplexers were exposed.

Table II summarizes and compares the FPGA voltage and the long-wire-coupling side-channel attacks.

C. System-Level Attacks

Attacks using cloud FPGAs can have impacts beyond the FPGA fabric. These devices can be manipulated to disclose information or generate faulty results from attached memory, caches, CPUs, and adjacent FPGAs. For example, cloud FPGAs can be programmed to fingerprint specific devices to disclose configurations of computing resources in the data center. Tian et al. [25] used a cloud FPGA to access PUFs implemented in DRAM. Distinctive DRAM decay patterns help distinguish specific FPGAs in the cloud. A similar goal was achieved using on-FPGA RO power wasters to create an identifiable per-FPGA voltage response [35]. Tian et al. [61] showed that PCIe contention could also be used to map the locations of cloud nodes. If one FPGA overuses the attached PCIe bus, an adjacent resource on the same bus suffers from excessive bus latency which is easily identifiable. Contention can thus effectively be used to map out the locations of the interconnected components (CPUs, memory, and FPGAs) in AWS EC2 F1 nodes. PCIe bus contention can also be used as a covert channel. Giechaskiel et al. [62] showed that a low-bandwidth channel can be established between two virtual machines (VMs) that use cloud FPGAs via PCIe bus contention. A VM can transfer a logic ‘1’ value to another VM that shares the bus by programming its FPGA to overuse the bus. The lack of contention indicates a logic ‘0’ transfer.

Cloud FPGAs have also been used to induce faults in attached DRAM and caches. Weissman et al. [63] showed that RowHammer [64] attacks could be efficiently executed on DRAM from an FPGA. Bit flips caused by the attack led to the exposure of an RSA encryption key. It was also shown in the paper that a cloud FPGA could attack the last level (LL) cache used by an attached CPU creating a covert side channel.

Table III summarizes and compares the research works on cloud system-level FPGA attacks.

TABLE II: Comparison of power side-channel and long-wire coupling attacks, together with thermal and BTI side channels. In bold, scenarios demonstrated on a public cloud.

Attack Objective	Type	Target circuit (the victim)	Malicious circuits (the attacker)	Evaluation platform
Recover the key	Intra FPGA	RSA (SPA attack [22]) AES (CPA attack [28], [51])	ROs as voltage sensors [22] TDC [28], [51]	Zynq-7020 (Zedboard) [22] Spartan-6 (Sakura-G) [28] **Virtex UltraScale+ (Amazon AWS)** [51]
Recover the key	Inter FPGA	AES (CPA attack [31])	TDC [31]	Spartan-6 (Sakura-G) [31]
Recover the key	Intra SoC	OpenSSL AES (CPA attack [52]) Tiny AES (CPA attack [52])	TDC [52]	Zynq-7000 [52]
Recover: DNN model [53], the architecture of NN layers [54], BNN inputs [30], folding parameters [55]	Intra FPGA	MLP [53], AlexNet [53], VGG16 [53] Versatile Tensor Accelerator (ResNet-18, MobileNet v1) [54] Convolution unit of a BNN [30] FINN-MLP with folding [55]	Three-stage ROs as voltage sensors [53] TDC [54], [30], [55]	Zynq-7000 SoC (Zedboard) [53] Zynq-7000 SoC (ZC706) [54] Artix-7 (ChipWhisperer) [30] Zynq UltraScale+ (ZCU1-4) [30] Virtex UltraScale+ [30] Pynq-Z1 (Z-7020 SoC)[55]
Recover the key	Intra FPGA	Long wire at the AES S-box input [23] Long wire carrying AES key [24]	Long wire side-channel leakage sensed with ROs [23], [24]	Cyclone IV E [23] Cyclone IV GX [23] Virtex 6 (ML605s) [24] Artix 7 (Digilent Nexys 4 DDR) [24] Artix 7(Digilent Basys 3) [24] Spartan 7 (ArtyS7) [24]
Covert communication	Thermal	Temperature sensors: RO sensors [16], [17]	Heaters: Power wasters [16], [17]	Stratix V [16] **SmartSSD (with an FPGA)** [17]
Recover previous user data	BTI	Long wires [19]	TDC [19]	ZCU102 Ultrascale+ [19] **Virtex UltraScale+ (Amazon AWS)** [19]

V. Remediation for Electrical-Level Attacks

A body of research investigated remediations against electrical-level cloud FPGA attacks. In this section, we introduce and discuss the proposed countermeasures, highlighting their key features. The strategies against system-level attacks are discussed shortly, in Sections VII and VIII.

A. Protection Against Fault-Injection and DoS Attacks

Approaches to address voltage-based fault injection and DoS attacks in multi-tenant FPGAs can be broken into two broad classes: bitstream scanning and run-time remediation. Scanning FPGA bitstreams or intermediate designs used to generate bitstreams can help identify potentially malicious logic structures such as ring oscillators. For example, Krautter et al. [65] and La et al. [66] have developed bitstream scanners that attempt to locate malicious circuits instantiated in a library. These circuits include ROs, self-clocked logic, high fanout circuits, and glitch amplifiers. Both tools regenerate a netlist from a partial bitstream and use graph-based algorithms to locate potentially malicious circuits. Although this approach can locate many types of circuits, it is, unfortunately, straightforward to build power wasters that have the same logical profile as legitimate circuits (Fig. 9g) [32]. Benign-looking constructs can be used to inject faults [40] or perform side-channel attacks [40]. This makes the job of such FPGA antivirus tools much harder. Another issue is the balance between false positives (benign designs which are red-flagged) and false negatives (malicious designs that escape detection).

During deployment, the software of the cloud service provider (CSP) should monitor the activities of various tenants and closely watch for electrical level issues [78]. For instance, by identifying suspicious tenants and reacting to voltage surges, a malicious tenant can be disabled and evicted from the FPGA fabric before causing harm and impacting other FPGA tenants [67]. However, the efficiency of such approaches can be greatly improved by providing proper support in the technology and toolchain of cloud FPGAs, allowing for quick de-configuration of malicious tenants. Runtime approaches for voltage attack detection typically involve the use of distributed voltage sensors [27]. Both Mirzargar et al. [79] and Provelengios et al. [36] use an array of low-overhead RO-based voltage sensors that can identify voltage droops. This information can be used for remediation. More recent work has shown that TDCs can identify droops more quickly, allowing for a faster response time [82].

Since voltage attacks can lead to fault injection (in a few microseconds) or board reset (in tens of microseconds), attack remediation techniques must be able to be rapidly deployed. Luo et al. [81] built a framework that controls the frequency of the target FPGA applications to avoid timing faults. Provelengios et al. [50] demonstrated a processor-based approach to suppress synchronous power wasters. Information from voltage sensors are transferred to an ARM core via a dedicated network. If an attack is detected, it is localized to a clock region, and the associated clock is deactivated, stopping the attack. The remediation approach was shown to suppress voltage attacks on a Stratix 10 FPGA within 20 microseconds, sufficiently fast to prevent board reset. In Nassar et al. [67], partial FPGA reconfiguration is used to disrupt the operation of loop-based ROs. The authors determined a configuration

TABLE III: Comparison of system-level attacks. In bold, attacks demonstrated on a public cloud.

Attack Objective	Type	Target (the victim)	Malicious circuits (the attacker)	Evaluation platform
Fingerprinting cloud FPGA instances [25]	FPGA-to-DRAM	Cloud infrastructure [25]	Decay-based DRAM PUF [25]	**Virtex UltraScale+ (Amazon AWS) [25]**
Retrieve cloud configuration; Reverse engineer the FPGA instance allocation algorithm	PCIe contention	Cloud infrastructure [61]	Remote user transferring data between CPU and FPGA, creating PCIe traffic [61]	**Virtex UltraScale+ (Amazon AWS) [61]**
Use PCIe traffic signatures for covert communication; Deduce cloud resource usage by monitoring PCIe bandwidth	Covert communication between virtual machines on FPGA-accelerated cloud instances; Side-channel leak of PCIe signatures of cloud users	Cloud infrastructure [62]	Remote user causing intensive PCIe traffic [62]	**Virtex UltraScale+ (Amazon AWS) [62]**
Covert communication between components powered by the same power supply unit	FPGA-to-FPGA CPU-to-FPGA GPU-to-FPGA	Cloud infrastructure [26]	Four-stage ROs (one inverter and three buffers) [26]	Kintex 7 (KC705) Artix 7 (AC701) Xeon E5645 CPU Xeon E5-2609 CPU Nvidia GeForce GPU [26]
Fault attack on RSA, leaking the private factors	FPGA-to-DRAM RowHammer	Shared DRAM [63]	RowHammer attacker from the FPGA [63]	Arria 10 GX PAC [63]
Cache-based covert communication	Cache attack FPGA-to-FPGA CPU-to-FPGA FPGA-to-CPU	CPU LL-cache, FPGA cache [63]	Software or hardware accessing the cache to evict, time, prime, or reload [63]	Arria 10 GX PAC [63]

sequence that was able to rapidly deactivate interconnect in a clock region of an UltraScale+ FPGA, effectively stopping a voltage attack. Partial reconfiguration was sufficiently fast enough to suppress a board crash and some timing faults.

B. Side-Channel Attack Remediation

Power SCAs can be defeated if preventive actions are deployed. FPGA SCA countermeasures can be classified into two categories: hiding and masking [83]. The goal of hiding is to reduce the signal-to-noise ratio of side channel information by increasing noise in the side channel or equalizing power consumption across computation [74]. Masking requires the processing of randomized data, while ensuring computation correctness [73]. Unfortunately, both approaches lead to area increases and the possibility of higher-order attacks [83]. For multi-tenant FPGAs, remediation approaches against power SCAs have been developed by Le Masle et al. [75] and Krautter et al. [76]. These approaches use a closed-loop control system to stabilize the steady-state power consumption of an FPGA circuit. In the former case, an on-chip RO network is used to monitor on-FPGA voltage [75]. A proportional-integral-derivative (PID) controller, whose PID constants are set so that the voltage measured by the sensors is kept approximately constant is used as a control circuit. The latter approach uses a fence composed of ROs between two neighboring FPGA tenants [76] to increase signal-to-noise ratio. The number of active ROs is controlled by a voltage sensor. The total size of the fence can be adjusted by the designer of the circuit to be protected. For an even more effective fence per unit of area, ROs can be combined with the abundant FPGA routing resources [77].

Effective remediation approaches against long-wire coupling prevent security-sensitive signals from being routed in the vicinity of other tenants' signals. Both computer-aided design (CAD) and architectural techniques have been developed to address the potential risks of crosstalk. Huffmire et al. isolated risky applications and their signals via moats and drawbridges [68]. Yazdanshenas and Betz proposed wrapping roles with wrappers made from FPGA logic [69]. All data transported to or from a role is encrypted. The approach leads to an 80% data transport latency increase and a 20% role area increase. A hardware isolation framework [70], [72] was developed which prevents security-critical nets from using long routing wires. The nets are isolated from other users' nets by routing them first and keeping subsequently routed signals away from them. For long wires that cannot fit within the design boundaries, wires surrounding sensitive signals are left unassigned. Seifoori et al. [71] modified PathFinder, an FPGA routing algorithm, to prevent potential crosstalk. Their approach requires the users to specify security-critical nets at design time and set parameters to control the use of wires adjacent to these nets.

Side-channels based on temperature and BTI effects are best addressed, first, by not allowing aggressive heating (a strategy effective also against targeted aging attacks aiming at FPGA impersonation [20]) and, second, by allowing sufficient recovery time between two subsequent FPGA tenants [16], [17], [19].

TABLE IV: Comparison of the proposed protections against electrical-level attacks on shared FPGAs.

	Applicability				Use of FPGA resources				Who should deploy them				
Protections	*crs*	*pwr*	*dos*	*flt*	*logic*	*wire*	*clk*	Type	*usr*	*vnd*	*csp*	Disclosure	Portable
Krautter et al. [65]	●	●	◐	◐	○	○	○	Passive	○	○	●	○*	N/A
La et al. [66]	●	◐	◐	◐	○	○	○	Passive	○	○	●	○*	N/A
Nassar et al. [67]	○	○	●	◐	●	●	●	Active	○	○	●	○	●
Huffmire et al. [68]	◐	○	○	○	●	●	○	Passive	●	○	○	○	○
Yazdanshenas and Betz [69]	◐	○	○	○	●	●	●	Passive	○	◐	●	●	●
Luo et al. [70]	●	○	○	○	○	●	○	Passive	○	●	○	○	●
Seifoori et al. [71]	●	○	○	○	○	●	○	Passive	○	●	○	○	●
Luo et al. [72]	●	○	○	○	○	●	○	Passive	○	●	○	●	●
Regazzoni et al. [73]	○	◐	○	○	●	●	○	Passive	●	○	○	○	●
Tiri et al. [74]	○	◐	○	○	●	●	○	Passive	●	○	○	○	●
Le Masle et al. [75]	○	◐	○	○	●	●	●	Active	●	○	○	○	○
Krautter et al. [76]	○	◐	○	○	●	●	●	Active	●	○	◐	○	○
Glamočanin et al. [77]	○	◐	○	○	●	●	●	Active	●	○	◐	○	○
Shen et al. [78]	○	○	○	●	●	●	●	Active	●	○	○	○	○
Provelengios et al. [36]	○	○	◐	●	●	●	●	Active	●	○	●	●	○
Mirzargar et al. [79]	○	○	◐	●	●	●	●	Active	○	○	●	●	●*
Stott et al. [80]	○	○	○	●	●	●	●	Active	●	○	○	○	○
Mahmoud et al. [41]	○	○	○	●	●	●	○	Active	●	○	○	○	○
Luo and Xu [81]	○	○	○	●	●	●	●	Active	●	○	○	○	○

Legend: *crs)* Crosstalk side-channel attack; *pwr)* Power side-channel attack; *dos)* Denial-of-service attack; *flt)* Fault attack; *logic)* 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.

C. Discussion

Table IV summarizes and compares proposed countermeasures, using the following criteria:

- Applicability, if a countermeasure is effective against more than a single attack type,
- Use of logic, wiring, and clock resources,
- Real-time (i.e., active) application or not real-time (i.e., passive),
- Deployed by users, FPGA vendors, or CSPs,
- Whether design disclosure to the CSP is a requirement for protection implementation, and
- Amount of effort required to port the countermeasure to an FPGA of another family or another vendor.

We can observe that countermeasures seldom target more than a single type of attack. Further, they often require additional FPGA resources. Some are easily portable between different FPGA families, but many are not. Clearly, combined efforts by researchers, end users, FPGA vendors, and CSPs are required to reach a more general solution, or at least a suitable combination of the existing ideas.

VI. Trends in Cloud System Use of FPGAs

Currently, the most widespread model of cloud-level use of FPGAs is the single node accelerator model, as discussed in Section II and illustrated in Fig. 2. In such a model, an FPGA accelerator node acts as a PCIe-attached co-processor, which remote users can access and program with designs via a host CPU. From the point of view of cloud service providers, this model is convenient for a number of reasons. First, CSPs can quickly and efficiently deploy an FPGA-accelerated cloud by using off-the-shelf boards, an approach used by Nimbix and Tencent [7]. Other CSPs, e.g., Amazon AWS, Baidu, Huawei, and Alibaba, have designed custom boards and tailored their hardware not only to specific user requirements, but also to their specific datacenter-level architecture requirements and upgrades. While custom board design implies higher start-up and maintenance costs, it provides the freedom of feature selection (e.g., FPGA family, size, I/O port count, and off-chip memory size and type). Another advantage of the distributed single node accelerator model is simplicity. FPGA instances are easier to orchestrate and the risk of their failure affecting a large number of datacenter resources is reduced. Additionally, in a single-node accelerator model, it is straightforward to offer FPGAs as bare-metal resources using standard FPGA-design tools.

Despite the aforementioned advantages, the single-node accelerator model is not here to stay. Future models for cloud system level use of FPGAs will need to provide higher scalability, minimum communication and data transfer latency, virtualization and sharing of FPGA resources, while addressing the accompanied security risks. In this section, we first discuss trends in FPGA-accelerator architectures, which address the challenge of scalability and latency. Then, we give a detailed overview of research on FPGA resource virtualization.

A. Trends in Cloud FPGA Architectures

Unlike production systems, research architectures are built with less concern regarding total implementation cost or security constraints, but rather focus on performance and latency. For instance, to achieve fast FPGA-to-FPGA communication, a number of research architectures deploy a secondary network which connects FPGAs across servers. Examples of such architectures include Microsoft's Catapult v1 [84], Novo-G# [85], Albireo nodes of the Cygnus supercomputer system at the University of Tsukuba, and the Noctua system at the Paderborn Center for Parallel Computing [86]. None of today's production systems use a secondary network, likely because of the cost and complexity of wiring and additional networking hardware and the resources needed to secure the system. However, designers of some production systems are considering providing fast FPGA-to-FPGA links; specifically, Amazon AWS is advertising its plans to enable FPGA cards to send or receive data from an adjacent card at 200 Gbps, over a generic raw streaming interface [87].

Another important trend among research architectures is to directly connect FPGAs to the datacenter network. Consequently, FPGAs could be accessed by a CPU or by another FPGA, leading to good scalability. Some examples of such architectures include CloudFPGA by IBM Zurich Research Lab [88], the University of Toronto SAVI testbed [89] (where a cluster of FPGAs is connected to the datacenter network), Enzian at ETH Zurich [90] (where an FPGA is connected to the network on one side and coherently attached to a server-class SoC on another side), and Microsoft Azure. Obviously, the security and reliability concerns inherent to direct network connectivity of FPGAs limit user FPGA access.

B. FPGA Resource Management and Virtualization

Users of cloud FPGA environments ideally want access to one or more dedicated FPGA boards without having to worry about resource sharing. However, cloud FPGA providers can sell FPGA services to more users if they can virtualize the environment and allow users who do not take up all FPGA and connected peripheral resources to share the underlying physical infrastructure. Sharing can be performed either by giving each user a specific time period in which to use the cloud environment, also called slot-based allocation, e.g., [91], or by allowing multiple users to use the environment at the same time (multitenancy). A virtualized view of peripherals can be provided to make it seem like each user has unique access, e.g., [92], [93], [94]. Multitenancy allows for increased flexibility and control.

The goal of a virtualized environment is to make it seem like a user has sole resource access, while physically serving several users at the same time. This approach for FPGAs is conceptually similar to traditional CPU virtualization which is widely deployed and understood. However, FPGA virtualization differs in several important ways that make existing virtualization solutions for CPUs unsuitable. One important difference is that FPGAs do not execute sequential programs one instruction at a time, but rather implement parallel circuits, so circuits from different users run simultaneously. Furthermore, FPGA circuits can contain asynchronous elements that are not directly controlled by an external clock, so stopping and restarting an FPGA is not feasible with the current technology. Since FPGAs are often used for time-sensitive processes, even if there was a way to interrupt an FPGA and restart it in the same internal state, it would likely interfere with application semantics.

To achieve a virtualized environment, external memory must be remapped to different physical addresses, and access to the FPGA must be carefully controlled to make sure each user cannot interfere with other users on the same hardware. The same is true for other external resources like the network or storage systems. Together a shell and the virtualized peripherals should create a good logical separation between different users, even if several users use the hardware at the same time. However, multiple problems have been presented in the last few years that cast doubt on the effectiveness of this separation technique [91], [95], [60], [24], [59], [15], [39].

The challenge of datacenter resource provisioning has motivated the development of platforms that allow datacenter managers to monitor the network and modify, in real-time, the amount and type of compute resources given to each application. Examples of such platforms are Openstack [96] (a free and open standard for cloud computing platforms) and Kubernetes [97]. While these platforms orchestrate the entire datacenter, they require individual components on each server to provision its resources. Furthermore, these platforms are currently limited to provisioning CPUs. For heterogeneous servers, extending and redesigning the existing orchestration platforms to include other types of computing components is a necessity. In particular, provisioning and sharing FPGA resources requires an entirely new solution for resource virtualization [98], operating system support, and FPGA programming (i.e., bitstream file creation and partial reconfiguration). Processor virtualization relies either on instruction set translation or hardware support with technologies such as Intel Virtualization Technology (Intel VT) [99]. Alternative approaches are needed for FPGAs. Two common approaches for FPGA resource management are slot-based allocation and FPGA overlays.

In slot-based FPGA resource management [7], [100], [101], [102], an FPGA is divided into several reconfigurable regions, in which user FPGA circuits can be mapped at runtime via reconfiguration. These regions may or may not be symmetric, i.e., use similar or identically sized slots. In an example of slot-based FPGA virtualization [103], an FPGA is divided into four regions, while the architecture multiplexes FPGAs by dynamically assigning resources. The main limitation of the architecture is the lack of on-chip communication between regions, resulting in a considerable data-copy overhead when such a transfer is required. To address the above issue and reduce data movement overhead, Mbongue et al. [104], [94] and Yazdanshenas and Betz [69] make use of the FPGA on-chip interconnect. A number of research proposals for FPGA virtualization architectures are listed in Table V. In addition to the solutions listed in the table, commercial solutions from VMAccell and InAccel offer complete frameworks. The VMAccel software is based on Openstack, Docker, and Kubernetes, while InAccel software contains high-level APIs in C/C++, Java, and Python and a unified engine to support a heterogeneous multi-accelerator platform.

FPGA overlays, also called intermediate architectures or fabrics [112], offer an alternative to FPGA partitioning. Overlays abstract away the low-level FPGA hardware components (e.g., LUTs, flip flops, DSPs, etc.). Higher-level coarse-grained

TABLE V: Trends in FPGA virtualization architectures.

	Access method	FPGA regions	Spatial sharing	On-chip comm.
Byma et al. [105]	10 GbE	✗	✓	✗
Chen et al. [103]	PCIe	4	✓	✗
Fahmy et al. [106]	PCIe	4	✓	✗
Weerasinghe et al. [107]	10 GbE	1	✗	✗
Asiatici et al. [108]	PCIe	3	✓	✗
Vesper et al. [109]	PCIe	4	✓	✓
Tarafdar et al. [110]	10 GbE	1	✗	✗
Zhang et al. [111]	PCIe	✗	✓	✗
Mbongue et al. [104]	PCIe	4	✓	✓
Mbongue et al. [92]	PCIe, Ethernet	6	✓	✓

processing elements, also called coarse-grained reconfigurable arrays (CGRAs), are implemented and can be programmed at runtime through software-level function calls. These elements are connected using interconnect topologies that allow both parallel processing and easy data exchange [7], [113], [114], [115], [116]. CGRAs are often supported by compilers that can map popular software programming languages. By abstracting away the low level hardware details, overlays also allow faster FPGA development cycles.

Virtualized FPGA hardware requires operating-system and hypervisor interaction. Previous work [7] has addressed the challenge of allowing spatial [105], [103], [117], [118], [119], [120], [101], [121], [122] and temporal multiplexing [103], [117], [118], [119], [123], [124], [121], [125], [122] of FPGA resources, and facilitating integration of FPGAs in datacenters and the cloud. The application of a traditional operating system resource abstraction to FPGAs has been recently explored by Korolija et al. [126]; the authors implemented a portable and configurable shell for FPGAs which supports secure spatial and temporal FPGA multiplexing, virtual memory, communication, and memory management inside an uniform execution environment. Optimus [127] is a hypervisor that supports scalable shared-memory FPGA virtualization, while offering spatial multiplexing of up to eight physical accelerators on a single FPGA, and temporal multiplexing to overprovision each of these accelerators. To isolate each guest's address space, Optimus uses the technique of page table slicing as a hardware-software co-design technique. Another example is FPGAVirt, which uses Virtio, an I/O virtualization framework initially implemented for Linux environments, to provide communication interfaces between the host virtual machine and FPGAs [104]. In FPGAVirt, an FPGA is abstracted away as an overlay architecture consisting of a two-dimensional array of routers and programmable processing elements, where each processing element is a virtual reconfigurable function. Zha and Li developed the ViTAL framework [121] that supports dynamic fine-grained resource management by abstracting heterogeneous resources of FPGA clusters into a homogeneous view of an array of virtual blocks and partitioning and mapping user applications onto those virtual blocks. In ViTAL, each block has the same type and amount of programmable resources and the same interface to the peripheral devices; furthermore, virtual blocks deployed on the same or different FPGAs use identical intra-block communication interfaces. The resulting illusion of a single and infinitely large FPGA reduces the programming complexity and enables scale-out acceleration. As a follow-up, Zha and Li developed Hetero-ViTAL [122] to address the challenges of heterogeneous FPGA clusters and demonstrate that adding a system abstraction as an indirection layer between application-specific instruction set architecture and hardware-specific abstractions substantially reduces resource management complexity [125].

Although scheduling is a well established technique for CPUs, state saving makes it a challenge for FPGAs. For preemptive scheduling, capturing and restoring the state of an accelerator can be prohibitively complex, because the accelerator state can be spread out across a large number of FPGA resources (LUTs, flip flops, BRAMs, etc.). Saving and restoring the state has been shown to take between microseconds to milliseconds, excluding partial reconfiguration latency [128]. In comparison, nonpreemptive scheduling is less costly, as the accelerators run to completion. Asiatici et al. [108] proposed dynamic scheduling that takes advantage of the free slots available at run-time to improve resource utilization and performance. Similar to nonpreemptive scheduling, cooperative scheduling can operate with minimal overhead, e.g., by offering context switching only when an accelerator reaches an execution checkpoint [129].

VII. Security Challenges for Next Generation Cloud FPGAs

The following section presents an overview of the security challenges likely to be faced due to current and expected trends in cloud FPGAs.

A. *Memory Timing and RowHammer*

When cloud data is remapped to a different part of the available physical memory, the user loses fine-grained control over where data is placed relative to other data blocks. Memory sharing and remapping inevitably changes the timing of memory access, which can lead to reduced performance, or in the worst case, allow a user to deduce information about other users in the system, e.g., [130]. This issue occurs because the activity of other users change the physical location of data in memory, and even if a physical address is hidden, it is still possible to use timing to deduce memory activity such as cache misses [131]. Such cache timing attacks have been used in the past to extract encryption keys and other secrets from memory [28], [31], [22]. It might also be possible to use RowHammer-like attacks [39], [63] to change a value in memory that a user is not permitted to access.

B. *Emerging Electrical Threats*

Given the virtualization of the FPGA fabric in the cloud and "FPGA as a Service" models, there are additional types of electrical vulnerabilities, threats and attacks, beyond those discussed in Section IV, that may arise. The cloud provider which rents the FPGA resources, needs to ensure that FPGA devices are not vandalized or improperly used. For instance, intentional over-aging may result in damage to the FPGA fabric or a significant reduction in the remaining useful lifetime of the FPGA. In the context of untrusted FPGA IP cores, FPGA Trojans may be able to perform electrical level attacks after a trigger input sequence is used. The trigger and payload parts of the Trojan could exploit electrical level vulnerabilities which are then also embedded in the IP core, making it stealthy and hard to detect.

Newer FPGAs (e.g., Xilinx Virtex UltraScale+ and Kintex UltraScale) are partitioned into super logic regions (SLRs), which are separate structures that are analogous to cores in a modern CPU. This isolation can assist virtualization and resource sharing, since each user can be assigned a single SLR, however it has been shown that it is possible to communicate between SLRs through power fluctuations, even if no direct wires connect the two regions [59]. Cross-SLR attacks are extremely hard to prevent because they utilize the physical properties of the underlying device (e.g., a shared PDN). This issue is likely to become more pronounced as the number of SLRs per device increase.

C. *Coordinated Attacks*

There are also new sets of vulnerabilities which may arise when the FPGA fabric is co-integrated with other cloud components such as CPUs and GPUs. This integration may lead to more powerful coordinated attacks in which the adversarial collaboration on both processor and FPGA sides may render existing countermeasures ineffective. Moreover, processes running on the processor or GPU may become new victims in such scenarios.

D. *System-Level Attacks*

The spectrum of system-level attacks and remediations that will affect cloud FPGAs is likely to follow the attacks present in current CPU-dominated systems. As FPGAs are integrated into the datacenter memory hierarchy, timing-based cache attacks in which specific instructions and data values are extracted from shared caches [132] are likely to increase. These types can be addressed in some cases by isolating cache addressing [133], limiting sharing among virtual machines that use FPGAs.

The use of bus contention as a covert channel is a concern for both CPU and FPGA compute elements. FPGAs may be used to extract machine learning [134] and encryption key values [135] from PCIe bus traffic between third parties, an approach previously used with GPUs and CPUs. New approaches to balance and mask bus traffic may be needed to prevent contention and snooping-based bus attacks.

Finally, FPGAs have been shown to be efficient in performing RowHammer attacks on DRAM. New approaches to isolate sensitive values in memory, similar to BlockHammer [136], may be used to separate global FPGA data from malicious activity by other users. However, an increase in interest in remote DMA and resource disaggregation may make the secure orchestration of memory locations more difficult [137].

VIII. Security techniques for next generation cloud FPGAs

To secure the next generation of FPGAs, designers should act at different layers and should use and combine several technologies. The hypervisor is the most natural module where security techniques and components, such as monitors, could be integrated and where security properties, such as isolation, could be verified and enforced. FPGAs are reconfigurable by nature. This provides designers with the flexibility to counteract side-channel attacks, but also to quickly address attacks targeting specific cryptographic algorithms by updating the algorithm implementation. Security in FPGAs can also be improved through changes to device architecture. These changes could make a device more resistant to certain attacks or make devices more suitable for cryptographic primitive implementation. Finally, the role of CAD tools on the security of cloud FPGAs should not be underestimated. CAD tools could potentially verify security properties or construct systems that are secure by design. However, if used without care, CAD tools could negatively affect security. The remainder of this section discusses the most promising techniques for securing the next generation of cloud FPGAs and their development.

A. *Hypervisor-Based Monitoring*

In a cloud setting in which resources are shared across multiple tenants, security and isolation are paramount. It is typically the duty of the cloud provider to ensure both features. One approach to providing security and isolation is to integrate monitoring functionality into a hypervisor. Monitor implementations include dedicated circuits and programmable modules that implement the needed security policies. In general, a monitor is a dedicated component (often a dedicated circuit, but it can be a software routine running on a microcontroller) that analyzes and promptly detects anomalous behavior that can be classified as malicious. In principle, monitors can be inserted in a system by designers or by the cloud provider. Designer-inserted monitors (sensors) in FPGAs would need to be managed carefully since the same structures used by some sensors (e.g., ROs) can also be used by attackers. In CSPs providing monitoring, the privacy of users must be ensured.

A step beyond the use of monitors is the development of complete hypervisor frameworks to provide isolation. For example, Hategekimana et al. [138] proposed a security framework to control the sharing of hardware modules in a heterogeneous cloud system composed of CPUs and FPGAs. The framework is derived from MAC-based hypervisors, and it is adapted to guarantee the isolation of hardware accelerators in shared FPGAs. The goal of the framework is to ensure that hardware monitors reside and are executed in the same security context of the virtual machine that "called" them. This goal is achieved by managing the privileges of the guest virtual machines at the software level.

B. *Hypervisor-based Remediation*

Despite the number of approaches addressing the challenge of FPGA virtualization and operating system support, current research lacks a comprehensive solution against the electrical-level attacks detailed in Section III. FPGA virtualization currently addresses physical isolation between tenants (spatial isolation within the reconfigurable fabric or memory address space isolation). However, as seen in Section III, isolation alone cannot prevent intra- and inter-chip electrical coupling. To that end, future hypervisors need at least the following three mechanisms: first, a mechanism to prevent (to the extent possible) a malicious design from being deployed in the cloud. Second, a mechanism to detect an ongoing electrical-level attack, and last but not least, a safe way to migrate the state of the FPGA accelerator that is potentially affected by the attack. The first mechanism requires, for instance, having the FPGA design software discard potentially malicious accelerators or forcing the user to create an alternative built with trusted primitives. The second mechanism requires the implementation of one or a combination of techniques detailed in Section V. Finally, the third mechanism requires the development of solutions for reliable and fast checkpointing, including error recovery, since hypervisor reaction time will inevitably be orders of magnitude longer than the time required by a fault-injection attack.

C. Use of FPGA Reconfiguration for Security

Reconfiguration is a powerful feature that FPGAs can use to ensure security. For example, reconfiguration is a useful tool for providing crypto-agility, which provides the capability of updating security primitives when they become obsolete or vulnerable to attacks that were not known when a system was deployed. This feature is clearly useful in the context of the cloud, especially when the deployment of not-yet-standardized algorithms is needed (as is the case of lightweight primitives or post-quantum algorithms). Reconfiguration has also been explored as a possible way to mitigate side channel attacks [139]. The principle on which this countermeasure is based is that it is hard to profile a device that keeps changing. This approach could also be a promising way to mitigate attacks in cloud settings. To ensure effective security using reconfiguration, the reconfiguration manager must not be compromised, since this action would allow the use of a malicious bitstream or the access to hardware resources by unauthorized software. To mitigate this problem, authentication modules paired with appropriate procedures for key management and challenge-response protocols for authentication could be used and communication to and from the hardware accelerators could be secured. It may be possible to rely on classical encryption [130] and access control mechanisms [140] to perform these actions. It is however necessary to ensure that they are resistant against physical attacks.

D. FPGA Architecture Enhancement

Many of the attacks on multi-tenant FPGAs are a result of the FPGA's shared PDN. Unlike multi-core microprocessors which typically have isolated power islands for each processor core [141], the PDN in individual FPGAs is not electrically isolated. Several research projects have examined allowing for tunable voltage for both logic and interconnect. Ahmed et al. [142] suggested optimizing the look-up table (LUT) design, to render its input-to-output delays less variable with the change of supply voltage. 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. Ebrahimi et al. [143] use a combination of hardened and reconfigurable logic to address changes in power consumption, as needed. Several projects have examined dynamically controlling the voltage for FPGA regions. Gayasan et al. [144] provided selectable voltages for the interconnect and logic in a logic cluster. More recent work examined voltage selection for regions of logic clusters [145]. Giechaskiel et al. described the possibility of isolating each super logic region on a separate PDN, although an implementation was not provided [59].

As mentioned in Section V, voltage sensing is a key component of voltage attack remediation. Although current FPGAs typically contain one (or a small number) of low sample rate hard voltage and thermal sensors [146], [147], more would be needed for a fast, reliable remediation strategy. Soft voltage sensors remain a viable option (e.g. Zick [27]), although they often have TDC structures that could be construed as malicious. AWS EC2 F1 currently employs an external power monitor [148] to identify power attacks consuming more than 80 W. Attack detection results in FPGA shutdown.

The use of sensors has limitations. If sensor data collection is supported by the shell, the trustworthiness of the shell becomes as issue. Even if the shell is trusted, the data collection and processing may take too much time to prevent the attack. Finally, the sensors themselves can be affected by the attack (e.g., a TDC can be decalibrated and recalibration can take an extended time).

Several architectural enhancements could improve an FPGA's ability to respond to a voltage attack. Nassar et al. [67] showed that partial reconfiguration can suppress an RO-based voltage attack in as little as 1.5 microseconds in an UltraScale+ FPGA. However, even faster dynamic reconfiguration approaches (e.g., a 'kill' signal) could be considered if an attack is detected. Although not used for fault suppression, Vipin et al. [149] developed a fast partial reconfiguration approach that could be used. Finally, FPGA communication could be isolated logically and electrically via network-on-chip (NoC) interconnection. Yazdanshenas and Betz [69] previously demonstrated this effective security approach.

Additional FPGA architectural changes can be considered to suppress SCAs. Recently, several approaches have attempted to reduce the amount of PDN information leakage. One idea is to use converter gating and distributed voltage regulators to reduce the amount of switching-dependent fluctuations on the PDN [150]. Other approaches use current flattening circuits against differential power analysis (DPA) attacks [151] or power profile scrambling [152], [153]. Such methods would extend existing hiding techniques (Section V-B). In general, it will be important to design PDNs with security constraints in mind. Although it is likely impossible to fully remove PDN information leakage, it can be suppressed to a certain level such that, together with solutions at higher abstraction levels, leakage is practically removed. In more advanced technology nodes, due to tighter wire pitch, the parasitic resistance and capacitance of wires increase, which in turn amplifies the amount of observed leakage through voltage fluctuations. This behavior further highlights the need for secure PDN design for cloud FPGAs.

At the design level, one can consider design styles which are inherently less susceptible to electrical level leakage. A promising solution is dual rail logic (DRL). It may be possible to provide proper circuit-level support to implement DRL efficiently in an FPGA fabric [154], [155], [156]. Since mapping DRL to an existing FPGA fabric does not allow for glitch-free design, it is necessary to redesign the FPGA fabric and design tools to allow for more efficient DRL realization of masked designs on FPGAs. In addition to the increased hardware design costs and associated performance and power overheads for mapped designs, toolchain compatibility is another concern. The effective realization of dual-rail logic and other masking schemes on FPGAs requires the support of the design automation and mapping tools. For instance, the two rails of the logic must be routed to minimize delay differences.

E. FPGA CAD Enhancements

FPGA mapping and physical design have a considerable impact on the amount of information leakage at the electrical level. An analysis [157] showed that the effect of physical design and mapping on the amount of information leakage between two tenants, measured in the number of traces needed to perform a correlation power analysis (CPA) attack on the AES implementation, could be more than 100×. This is both good news and bad news, since many countermeasures have the same level of effectiveness. As a result, some countermeasures could be almost nullified by ignoring the effect of physical design and wrongly mapping a trusted (victim) tenant in a very sensitive region of the fabric. However, by carefully choosing the region and physical design of the victim tenant and the floorplanning and placement of the potentially-malicious tenants, more than 100× protection can be achieved at no extra hardware costs (including online monitoring, wrapper circuitry, etc.).

This highlights the importance of FPGA CAD on suppressing electrical-level information leakage. One challenge which cannot be fully ignored is the impact of chip-to-chip variations. As a result, the final mapping of the victim and untrusted tenants should be fine-tuned to the specific FPGA board. Additionally, the design of proper wrappers (around the victim tenants) and sandboxing (around the untrusted tenants) should be automated and included as a part of a secure FPGA mapping flow. This further highlights the complexity of such attacks and potential countermeasures, given various dependencies on the respective placement of the victim's and attacker's blocks, as well as the specific boards, which can relatively increase (or reduce) the attack and countermeasure efficiency multiple fold. Such still-open research challenges motivate further research on this topic to find suitable solutions.

F. Trust in FPGAs and their components

FPGAs largely rely on IP cores for the development of complete systems. This approach is also used for systems deployed in the cloud. The use of third party IP cores (which, in cloud FPGAs, is even more common than in standalone reconfigurable devices) brings several challenges related to the trust of components and the entities involved in the design and deployment chain. A model of trust for current cloud-based FPGAs is summarized by Turan and Verbauwhede [158]. The model considers three main entities: the platform provider, the accelerator developer (who develops accelerators for specific tasks), and the application developer. The platform provider trusts neither the accelerator developer nor the application developer. The accelerator developer, instead, is required to trust the platform provider (for instance, providing the accelerators in a non-encrypted form). In this scenario, a malicious platform provider can comprehend the IPs created by the accelerator provider. Finally, the application provider must trust both the platform and accelerator providers. Among the model limitations, the authors report that only platform providers are protected.

To address the limitations of the current model, it is necessary to include mechanisms to protect IP providers from piracy or other similar illegitimate use of their IPs. Common IP protection methods proposed in the literature involve encryption at the bitstream level [158]. For these methods to be successfully ported to future cloud FPGAs, challenges such as cryptographic key management and simulation and debugging of the interoperation of the encrypted IP bitstreams with other hardware blocks will need to be addressed [158]. Some published works rely on a trusted third party (TTP) [159], [160], [161], [162]. Turan and Verbauwhede [158] argue that, in the cloud FPGA context, TTPs could be involved as entities responsible for cryptographic key management. Additionally, having TTPs take a share of the license fee of each IP core via a pay-per-use licensing scheme could further incentivize TTPs to invest in protecting the IP cores they offer.

The importance of IP protection has motivated the rise of start-ups. An example is Accelize, whose business model involves designing custom accelerators for customers and supporting third-party developers to offer their accelerators to Accelize clients. The protection is achieved using a proprietary Digital Rights Management (DRM) solution, compatible with various FPGA accelerated cloud platforms. Their DRM wraps the IPs, protects them (via a licensing scheme), and meters the IP use. The obvious downsides are the required trust in a third party (which is Accelize itself) and the added cost for end users.

G. Single Tenancy vs Multitenancy

Considering electrical-level attacks enabled by FPGA multitenancy, an alternative strategy would be to run each tenant on a separate FPGA. However, resorting to this extreme policy would erode most of the gains from FPGA cloud virtualization and significantly increase upfront investment in the FPGA fleet. Smaller FPGAs would incur more costs at the board level and create a communication bottleneck, which may not be economical or high-performing. As a last security resort, some users might opt to have their own FPGA units and avoid multitenancy. Therefore, hybrid FPGA fleets containing many large and high-performance FPGAs for multitenancy and a limited selection of smaller FPGAs for single-use for security reasons might be favorable.

IX. Lessons Learned and Next Steps

In the previous sections, we discussed various electrical-level attacks for cloud FPGAs and countermeasures to deal with them. Here we summarize some of the key takeaways from the research performed over the past several years on this topic, and provide some insights for the next generation of secure FPGA platforms to be deployed in cloud computing.

A. Lessons Learned

Our main takeaways can be grouped into three topics: leakage, mitigation mechanisms, and deployment updates. In this section, we discuss each in turn.

Electrical-Level Leakage is Unavoidable. Because FPGAs, and indeed any integrated circuit, have a common electrical medium throughout the chip, electrical-level leakage will always be unavoidable to some extent. This problem is made worse by the shared PDN in many integrated circuits. Such fundamental electrical relationships between different parts of a chip inherently undermine any attempt at isolation at the logic level and above.

On cloud FPGAs, electrical-level attacks are made more difficult in the presence of additional activity by other users and the shell, but with more samples and post processing, they remain feasible. Varying victim and adversary locations on the FPGA have generally failed to completely isolate different users from each other. Providing complete isolation at the electrical level is, if not impossible, impractical and extremely costly. *A certain level of information leakage should be the underlying assumption for all security solutions for cloud FPGAs.*

Mitigation Mechanisms Are Required Across All Levels. Since electrical-level leakage is fundamentally unavoidable, design-level solutions for electrical-level isolation are also incomplete and cannot fully suppress the leakage. However, despite the fact that the problem cannot be entirely eliminated, there is a need for run-time solutions that predict, detect, and mitigate electrical-level attacks. This could be in the form of treating important data in such a way that leakage has minimal impact, e.g., avoid transferring encryption keys between different parts of a design, etc. *There is a need for a holistic cross-layer approach for secure FPGA platforms from design to deployment.*

Continuous Updates of Deployed Measures are Required. New electrical-level attacks are continuously being found and existing attacks have become more stealthy, evading existing countermeasures. Often new attacks are not fundamentally undetectable, they just evade existing countermeasures. This point highlights the importance of continuously updating a deployed system to ensure that it stays resistant to new attacks. *There is a constant need to update coordinated countermeasures at all levels of design and deployment.*

Below are some of the specific insights from existing research on cloud FPGA vulnerabilities and countermeasures:

- Side channel attacks can be made harder in the presence of noise, which is the basis for *hiding* countermeasures. Noise can be deliberately generated by the victim (to protect itself). However, the presence of noise can be outside the control of the victim or the adversary, as it could be caused by other accelerators or the shell. Due to this noise, sensor readings may be misleading (both for side-channel attacks and attack detection/mitigation). Also, using sensors to control noise generators (e.g., a noisy fence) can be suboptimal given that sensors pick up the voltage variations caused by all the activity on the chip.
- Noise generators (as part of a fence) have to be implemented carefully to ensure that the added noise does not destabilize the PDN, making it more vulnerable to unwanted (benign) reliability faults in a co-tenant or making the tenant design more vulnerable to fault attacks.
- FPGA power wasters can be misused to inject computational faults in the CPU or other components on the same SoC, so protections must be extended to other components besides the shell/logic. By using the common PCIe bus and other system interfaces and buses, fault injection and side channel attacks originating from the FPGA can affect other system components.
- Isolating shell logic and wiring is mandatory, given that tapping into wires (e.g., via crosstalk) could reveal secrets.
- The techniques and strategies used to provide security may vary greatly, depending on whether or not multitenancy is supported. Even without multitenancy, there is still leakage between the processor and the other parts of the system (CPU, GPU, network) through the PCIe bus, enabled by the common PDN, which should be mitigated.

B. Next steps

To tackle these new FPGA security challenges for cloud usage, and ensure secure and efficient FPGA virtualization in the cloud, there is a need for a set of orchestrated solutions. These solutions span a spectrum from the design and fabrication of a secure FPGA fabric, to the secure mapping of user designs to FPGAs using a mapping toolchain, secure FPGA CAD tools, and hypervisor deployment strategies. Thus, the fabric and toolchain for secure cloud FPGAs might look very different from today's systems. Solutions should also consider fabrication (secure FPGA fabric against electrical level attacks), design (the design of primitives and wrappers to mitigate information leakage and fault attacks), mapping (automating the modular design of secure designs to ensure proper isolation at the electrical level) and deployment (proper offline and runtime mechanisms by the hypervisor).

Given the nature and medium of electrical-level leakage, it is imperative to secure the FPGA fabric against electrical-level attacks. PDNs should be designed with security constraints in mind. One approach is to suppress information leakage at the PDN level. Although it may be impossible to fully remove PDN information leakage, it may be suppressed to a certain level such that together with higher abstraction levels, leakage

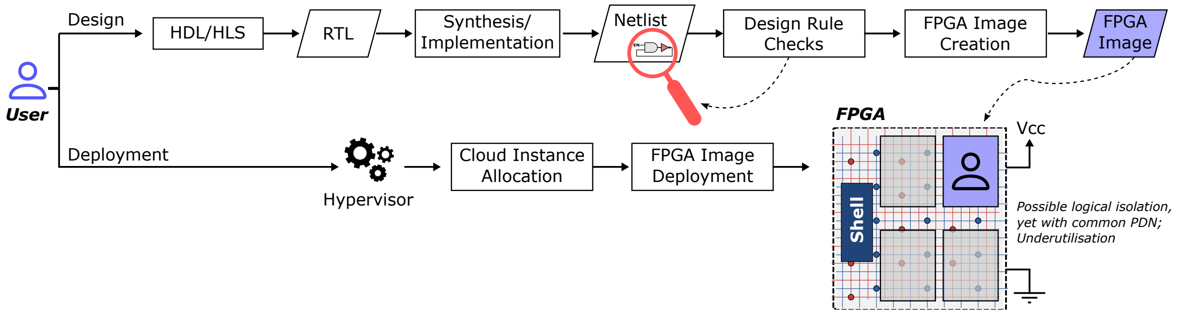


Fig. 10: Today's solutions for design deployment on cloud FPGAs. Even though the tools can achieve logical isolation between multiple tenants, due to electrical-level security issues discussed in this manuscript, the entire FPGA gets allocated for one user.

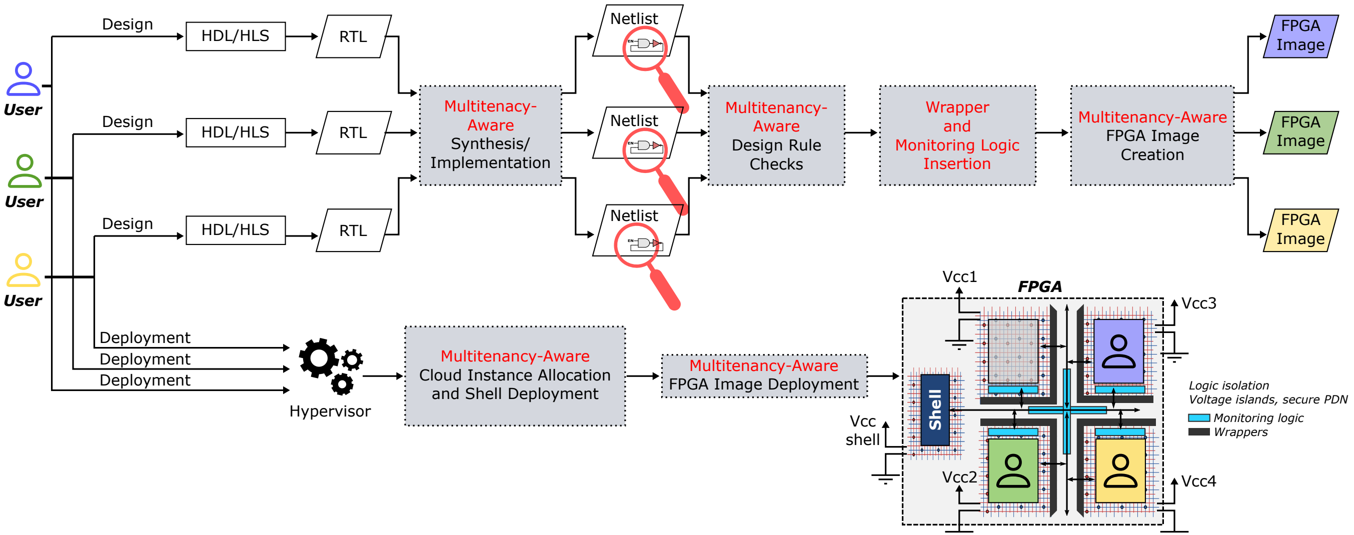

User
User
User
Design
Design
Design
HDL/HLS
HDL/HLS
HDL/HLS
RTL
RTL
RTL
Multitenacy-Aware Synthesis/ Implementation
Netlist
Netlist
Netlist
Multitenancy-Aware Design Rule Checks
Wrapper and Monitoring Logic Insertion
Multitenancy-Aware FPGA Image Creation
FPGA Image
FPGA Image
FPGA Image
Deployment
Deployment
Deployment
Hypervisor
Multitenancy-Aware Cloud Instance Allocation and Shell Deployment
Multitenancy-Aware FPGA Image Deployment
Vcc1
FPGA
Vcc3
Shell
Vcc shell
Vcc2
Vcc4
Logic isolation
Voltage islands, secure PDN
Monitoring logic
Wrappers

Fig. 11: Tomorrow's solutions for design deployment on multi-tenant cloud FPGAs, taking three tenants as an example. To alleviate electrical-level security issues, existing tools will need to be adapted and new steps (e.g., monitoring logic and wrapper insertion) will likely need to be added. New FPGA fabrics with reduced power side-channel coupling (e.g., with voltage islands) will need to be developed. Cloud FPGA shells will need to handle security-related tasks as well.

is practically removed. Another approach is to provide voltage islands for separate tenants on the same FPGA fabric. In more advanced technology nodes, due to tighter wire pitch, the parasitic resistance and capacitance of wires increase, which in turn amplifies the amount of observed leakage through voltage fluctuations. This issue further highlights the need for secure PDN design for cloud FPGAs. This approach, of course, comes with extra overheads in terms of routing, chip area, and potentially delay, power, and design closure.

Design automation toolchains for FPGAs should become aware of such electrical-level vulnerabilities and should support automatic analysis and identification of malicious constructs and potential leakage as well as integration of proper countermeasures and wrappers into the FPGA designs. An extension of the signal integrity analysis check, with all post-layout parasitics (PAX), may be required to analyze the potential leakage and embed proper countermeasures in the design to counter it. Of course, this adds extra complexity to the overall FPGA design and mapping toolchain, and clever solutions are required for tractable design closure and sign-off.

Proper protection requires both offline and online methods. Tenant designs must be certified against known malicious behaviors before being loaded into cloud FPGAs. This certification includes hypervisor checking of the tenant design and bitstream, and potentially the source register-transfer level (RTL), against known malicious activities and constructs. This activity includes both static (structural) checks and dynamic checks using accurate timing simulation. Due to stealthy nature of such malicious constructs, machine-learning approaches that automatically learn and generalize offline countermeasures and machine learning-based anomaly detection approaches executed at runtime that can predict and prevent attacks seem very promising.

There likely will be new vulnerabilities and associated attack vectors related to FPGA deployment in cloud settings. In the context of untrusted FPGA IP cores, the issue of FPGA Trojans performing electrical-level attacks should also be considered. The trigger and payload parts of the Trojan could exploit electrical-level vulnerabilities that are embedded in the IP core, making them stealthy and hard to detect.

Vulnerabilities may arise when the FPGA fabric is co-integrated with other cloud components, such as microprocessors and GPUs. This integration may lead to powerful coordinated attacks in which adversarial collaboration on both processor and FPGA sides may render existing countermeasures ineffective. Moreover, processes running on the processor or GPU sides may become new victims in such scenarios.

Figs. 10 and 11 illustrate today's and, the way we see them, tomorrow's solutions for design deployment on multi-tenant cloud FPGAs. We envision numerous changes, affecting many steps of FPGA manufacturing, compilation, and deployment.

C. Is It Worth It?

Last but not least, the question arises as to whether all the hardware design and manufacturing efforts, toolchain redesign efforts, hypervisor costs and extra performance, and area and power penalties associated with security measures for cloud FPGAs are worth it? The answer is the benefit of sharing. Providing true multitenancy and sharing of virtualized FPGA resources in the cloud, which is now hindered by the security concerns, can unleash the benefits of reconfigurable cloud computing. This effort would enable flexibility, increased performance and cost-efficiency for all types of users, no matter how large a fabric they require, and allow the cloud provider to reach out to a wider range of users and use cases.

X. Summary

In this manuscript, we provide a visionary look at the security issues associated with the diffusion and use of reconfigurable cloud computing. By critically reviewing successfully demonstrated remote FPGA attacks, we have shown and demonstrated the severity and scale of the threat. It is evident that current attacks are capable of undermining the availability of resources, the integrity of applications running on top of them, and the confidentiality of application data.

Attacks that have been successfully demonstrated so far have either targeted the FPGA itself or have used an FPGA to assist in system-level attacks. The former leverages electrical coupling between the adversary and a victim to pick up side-channel information or to generate a disturbance that injects faults via the electrical medium. The latter corrupts portions of memory shared with CPUs or overstresses shared components. These attacks generally require the use of on-chip voltage sensors to measure side channels, and power-wasting circuits to inject faults. Both components can be easily implemented in FPGAs due to the low-level hardware control and bit-level programmability that is offered. Sensors can be counteracted using bitstream scanning which identifies malicious design circuits. Dynamic attacks can be addressed with run-time remediation, for instance, the use of a closed-loop control system to stabilize steady-state power consumption in an effort to mitigate side-channel leakage.

From our study, it appears evident that the next generation of reconfigurable cloud computing security will require designers to consider multiple different operation levels and combine multiple technologies. At the system level, the most natural module to enforce security policies is the hypervisor, which should provide and guarantee isolation. At the same time, security improvements are expected to also come from architectural enhancements that could make FPGAs more resistant to attacks, and from CAD tools, whose role in security is often underestimated.

It is, however, necessary to acknowledge that electrical-level leakage is unavoidable. This leakage is intrinsic to FPGAs that share an on-chip electrical medium. As a result, designers should be aware that a certain level of information leakage is present for all cloud FPGA security solutions. Such awareness should be the guiding assumption when designing mitigation mechanisms that should necessarily be tackled in a holistic manner and allow for continuous updates to address the evolving attack surface.

References

[1] *Cloud Tensor Processing Units (TPUs)*, Google, 2023. [Online]. Available: https://cloud.google.com/tpu/docs/tpus

[2] *Project Catapult*, Microsoft Research, 2019. [Online]. Available: https://www.microsoft.com/en-us/research/project/project-catapult/

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

[4] *Compute Optimized Instance Families with FPGAs*, Alibaba, 2023. [Online]. Available: alibabacloud.com/help/doc-detail/108504.htm

[5] *Machine Learning*, Microsoft Azure. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/machine-learning/

[6] *Zynq UltraScale+ MPSoC*, Xilinx, 2023. [Online]. Available: xilinx.com

[7] C. Bobda, J. M. Mbongue, P. Chow, M. Ewais, N. Tarafdar, J. C. Vega, K. Eguro, D. Koch, S. Handagala, M. Leeser, M. Herbordt, H. Shahzad, P. Hofste, B. Ringlein, J. Szefer, A. Sanaullah, and R. Tessier, "The future of FPGA acceleration in datacenters and the cloud," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 15, no. 3, pp. 34:1 – 34:42, Sep. 2022.

[8] *Brainwave Project*, Microsoft, 2022. [Online]. Available: https://www.microsoft.com/en-us/research/project/project-brainwave/

[9] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in Science & Engineering*, vol. 12, no. 3, pp. 66–73, May 2010.

[10] A. Koneru, A. Todri-Sanial, and K. Chakrabarty, "Reliable power delivery and analysis of power-supply noise during testing in monolithic 3D ICs," in *37th VLSI Test Symposium (VTS)*, Apr. 2019, pp. 1–6.

[11] S.-C. Hung and K. Chakrabarty, "Design of a reliable power delivery network for monolithic 3D ICs," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Mar. 2020, pp. 1746–1751.

[12] S. Lin and N. Chang, "Challenges in power-ground integrity," in *2001 IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2001, pp. 651–654.

[13] N. Evmorfopoulos, D. Karampatzakis, and G. Stamoulis, "Precise identification of the worst-case voltage drop conditions in power grid verification," in *2006 IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2006, pp. 112–118.

[14] 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*, Sep. 2019, pp. 202–210.

[15] I. Giechaskiel, K. Rasmussen, and J. Szefer, "Measuring long wire leakage with ring oscillators in cloud FPGAs," in *Proceedings of the 29th International Conference on Field-Programmable Logic and Applications*, Sep. 2019, pp. 45–50.

[16] S. Tian and J. Szefer, "Temporal thermal covert channels in cloud FPGAs," in *Proceedings of the 27th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2019, pp. 298–303.

[17] T. Trochatos, A. Etim, and J. Szefer, "Security evaluation of thermal covert-channels on SmartSSDs," May 2023, arXiv: 2305.09115.

[18] S. Mahapatra, N. Goel, S. Desai, S. Gupta, B. Jose, S. Mukhopadhyay, K. Joshi, A. Jain, E. A. Islam, and M. A. Alam, "A comparative study of different physics-based NBTI models," *IEEE Transactions on Electron Devices*, vol. 60, no. 3, pp. 901–916, Mar. 2013.

[19] C. Drewes, O. Weng, A. Meza, A. Althoff, D. Kohlbrenner, R. Kastner, and D. Richmond, "Pentimento: Data remanence in cloud FPGAs," Mar. 2023, arXiv: 2303.17881.

[20] H. Cook, J. Thompson, Z. Tripp, B. Hutchings, and J. Goeders, "Cloning the unclonable: Physically cloning an FPGA ring-oscillator PUF," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2022, pp. 1–10.

[21] M. Zhao, M. Gao, and C. Kozyrakis, "ShEF: Shielded enclaves for cloud FPGAs," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2022, pp. 1–16.

[22] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 805–820.

[23] 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 *Proceedings of the 26th IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr. 2018, pp. 45–52.

[24] I. Giechaskiel, K. Eguro, and K. Rasmussen, "Leakier wires: Exploiting FPGA long wires for covert- and side-channel attacks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 3, pp. 1–29, Sep. 2019.

[25] S. Tian, W. Xiong, I. Giechaskiel, K. Rasmussen, and J. Szefer, "Fingerprinting cloud FPGA infrastructures," in *Proceedings of the 28th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2020, pp. 58–64.

[26] I. Giechaskiel, K. Rasmussen, and J. Szefer, "C^{3}APSULe: Cross-FPGA covert-channel attacks through power supply unit leakage," in *IEEE Symposium on Security and Privacy (SP)*, May 2020, pp. 1728–1741.

[27] 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*, Feb. 2013, pp. 101–104.

[28] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," *IEEE Design & Test*, vol. 38, no. 3, pp. 58–66, Jun. 2021.

[29] 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*, Mar. 2020, pp. 1007–1010.

[30] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Remote power side-channel attacks on BNN accelerators in FPGAs," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Feb. 2021, pp. 1639–1644.

[31] F. Schellenberg, D. R. 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*, Nov. 2018, pp. 1–7.

[32] 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*, Sep. 2020, pp. 231–235.

[33] 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*, Sep. 2017, pp. 1–7.

[34] K. Matas, T. M. La, K. D. Pham, and D. Koch, "Power-hammering through glitch amplification—attacks and mitigation," in *Proceedings of the 28th IEEE Symposium on Field-Programmable Custom Computing Machines*, May 2020, pp. 65–69.

[35] T. La, K. D. 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. 441–464, 2021.

[36] G. Provelengios, D. Holcomb, and R. Tessier, "Characterizing power distribution attacks in multi-user FPGA environments," in *Proceedings of the 29th International Conference on Field-Programmable Logic and Applications*, Sep. 2019, pp. 194–201.

[37] 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*, Mar. 2019, pp. 1745–1750.

[38] D. G. Mahmoud, D. Dervishi, S. Hussein, V. Lenders, and M. Stojilović, "DFAulted: Analyzing and exploiting CPU software faults caused by FPGA-driven undervolting attacks," *IEEE Access*, vol. 10, pp. 134 199–134 216, Dec. 2022.

[39] J. Krautter, D. R. Gnad, and M. B. Tahoori, "FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 3, pp. 44–68, Aug. 2018.

[40] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "Remote and stealthy fault attacks on virtualized FPGAs," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Feb. 2021, pp. 1632–1637.

[41] D. G. Mahmoud, W. Hu, and M. Stojilović, "X-Attack: Remote activation of satisfiability don't-care hardware Trojans on shared FPGAs," in *Proceedings of the 30th International Conference on Field-Programmable Logic and Applications*, Aug. 2020, pp. 185–92.

[42] G. Provelengios, D. E. Holcomb, and R. Tessier, "Power distribution attacks in multitenant FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 12, pp. 2685–2698, Dec. 2020.

[43] X. Li, R. Tessier, and D. Holcomb, "Precise fault injection to enable DFIA for attacking AES in remote FPGAs," in *Proceedings of the 30th IEEE Symposium on Field-Programmable Custom Computing Machines*, May 2022, pp. 1–5.

[44] 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 *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, Aug. 2019, pp. 48–55.

[45] A. Boutros, M. Hall, N. Papernot, and V. Betz, "Neighbors from hell: Voltage attacks against deep learning accelerators on multi-tenant FPGAs," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2020, pp. 103–11.

[46] Y. Luo, C. Gongye, Y. Fei, and X. Xu, "DeepStrike: Remotely-guided fault injection attacks on DNN accelerator in cloud-FPGA," in *58th ACM/ESDA/IEEE Design Automation Conference*, Jun. 2021, pp. 295–300.

[47] A. S. Rakin, Y. Luo, X. Xu, 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. 1919–1936.

[48] D. R. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "Analysis of transient voltage fluctuations in FPGAs," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2016, pp. 12–19.

[49] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "An experimental evaluation and analysis of transient voltage fluctuations in FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 10, pp. 1817–1830, 2018.

[50] G. Provelengios, D. E. Holcomb, and R. Tessier, "Mitigating voltage attacks in multi-tenant FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 14, no. 2, pp. 9:1–9:24, Jul. 2021.

[51] O. Glamočanin, D. G. Mahmoud, F. Regazzoni, and M. Stojilović, "Shared FPGAs and the Holy Grail: Protections against side-channel and fault attacks," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Feb. 2021, pp. 1–6.

[52] 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)*, Nov. 2019, pp. 109–125.

[53] 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. 4377–4388, Aug. 2021.

[54] 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*, May 2021, pp. 242–246.

[55] 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*, May 2022, pp. 1–10.

[56] D. R. E. Gnad, V. Meyers, N. M. Dang, F. Schellenberg, A. Moradi, and M. B. Tahoori, "Stealthy logic misuse for power analysis attacks in multi-tenant FPGAs," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Feb. 2021, pp. 1012–15.

[57] B. Udugama, D. Jayasinghe, H. Saadat, A. Ignjatovic, and S. Parameswaran, "VITI: A tiny self-calibrating sensor for power-variation measurement in FPGAs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 1, pp. 657–678, 2022.

[58] D. R. E. Gnad, C. D. K. Nguyen, S. H. Gillani, and M. B. Tahoori, "Voltage-based covert channels using FPGAs," *ACM Transactions on*

Design Automation of Electronic Systems (TODAES), vol. 26, no. 6, pp. 1–25, Nov. 2021.

[59] I. Giechaskiel, K. Rasmussen, and J. Szefer, "Reading between the dies: Cross-SLR covert channels on multi-tenant cloud FPGAs," in *37rd IEEE International Conference on Computer Design (ICCD)*, Nov. 2019, pp. 1–10.

[60] I. Giechaskiel, K. Rasmussen, and K. Eguro, "Leaky wires: Information leakage and covert communication between FPGA long wires," in *Proceedings of 13th ACM ASIA Conference on Information, Computer and Communications Security (ASIACCS)*, May 2018, pp. 15–27.

[61] S. Tian, I. Giechaskiel, W. Xiong, and J. Szefer, "Cloud FPGA cartography using PCIe contention," in *Proceedings of the 29th IEEE Symposium on Field-Programmable Custom Computing Machines*, 2021, pp. 224–232.

[62] I. Giechaskiel, S. Tian, and J. Szefer, "Cross-VM information leaks in FPGA-accelerated cloud environments," in *International Symposium on Hardware-Oriented Security and Trust*, Dec. 2021, pp. 91–101.

[63] Z. Weissman, T. Tiemann, D. Moghimi, E. Custodio, T. Eisenbarth, and B. Sunar, "JackHammer: Efficient rowhammer on heterogeneous FPGA-CPU platforms," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, p. 169–195, Jun. 2020.

[64] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *International Symposium on Computer Architecture*, Jun. 2014, p. 361–372.

[65] J. Krautter, D. R. Gnad, and M. B. Tahoori, "Mitigating electrical-level attacks towards secure multi-tenant FPGAs in the cloud," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 3, pp. 1–26, 2019.

[66] 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, pp. 1–31, Sep. 2020.

[67] H. Nassar, H. AlZughbi, D. R. E. Gnad, L. Bauer, M. B. Tahoori, and J. Henkel, "LoopBreaker: Disabling interconnects to mitigate voltage-based attacks in multi-tenant FPGAs," in *2021 IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2021, pp. 1–6.

[68] T. Huffmire, B. Brotherton, N. Callegari, J. Valamehr, J. W. R. Kastner, and T. Sherwood, "Designing secure systems on reconfigurable hardware," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 13, no. 3, pp. 44:1–44:24, Jul. 2008.

[69] S. Yazdanshenas and V. Betz, "The costs of confidentiality in virtualized FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 10, pp. 2272–2283, Oct. 2019.

[70] Y. Luo and X. Xu, "HILL: A hardware isolation framework against information leakage on multi-tenant FPGA long-wires," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2019, pp. 12–19.

[71] Z. Seifoori, S. S. Mirzargar, and M. Stojilović, "Closing leaks: Routing against crosstalk side-channel attacks," in *Proceedings of the 28th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2020, pp. 197–203.

[72] Y. Luo, S. Duan, and X. Xu, "FPGAPRO: A defense framework against crosstalk-induced secret leakage in FPGA," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 27, no. 3, pp. 1–31, May 2022.

[73] F. Regazzoni, Y. Wang, and F.-X. Standaert, "FPGA implementations of the AES masked against power analysis attacks," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, Feb. 2011, pp. 56–66.

[74] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Feb. 2004, pp. 1–6.

[75] 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*, Jan. 2011, pp. 1–8.

[76] J. Krautter, D. R. 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*, Nov. 2019, pp. 1–8.

[77] O. Glamočanin, A. Kostić, S. Kostić, and M. Stojilović, "Active wire fences for multitenant FPGAs," in *Proceedings of the 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, May 2023, pp. 1–8.

[78] L. L. Shen, I. Ahmed, and V. Betz, "Fast voltage transients on FPGAs: Impact and mitigation strategies," in *Proceedings of the 27th IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr. 2019, pp. 271–279.

[79] S. S. Mirzargar, G. Renault, A. Guerrieri, and M. Stojilović, "Nonintrusive and adaptive monitoring for locating voltage attacks in virtualized FPGAs," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2020, pp. 588–589.

[80] E. Stott, J. M. Levine, P. Y. K. Cheung, and N. Kapre, "Timing fault detection in FPGA-based circuits," in *Proceedings of the 22nd IEEE Symposium on Field-Programmable Custom Computing Machines*, May 2014, pp. 96–99.

[81] Y. Luo and X. Xu, "A quantitative defense framework against power attacks on multi-tenant FPGA," in *2020 IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2020, pp. 1–4.

[82] S. Moini, X. Li, P. Stanwicks, G. Provelengios, W. Burleson, R. Tessier, and D. Holcomb, "Understanding and comparing the capabilities of on-chip voltage sensors against remote power attacks on FPGAs," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2020, pp. 941–944.

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

[84] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A reconfigurable fabric for accelerating large-scale datacenter services," in *International Symposium on Computer Architecture*, Jun. 2014, pp. 1–12.

[85] A. D. George, M. C. Herbordt, H. Lam, A. G. Lawande, J. Sheng, and C. Yang, "Novo-g#: Large-scale reconfigurable computing with direct and programmable interconnects," in *IEEE High Performance Extreme Computing Conference (HPEC)*, Sep. 2016, pp. 1–7.

[86] C. Plessl, "Bringing FPGAs to HPC production systems and codes," in *Fourth International Workshop on Heterogeneous High-Performance Reconfigurable Computing*, Nov. 2018.

[87] *FPGA Link*, AWS GitHub, 2022. [Online]. Available: https://github.com/HFTrader/aws-fpga/blob/master/FAQs.md

[88] F. Abel, J. Weerasinghe, C. Hagleitner, B. Weiss, and S. Paredes, "An FPGA platform for hyperscalers," in *IEEE 25th Annual Symposium on High-Performance Interconnects (HOTI)*, Aug. 2017, pp. 29–32.

[89] T. Lin, B. Park, H. Bannazadeh, and A. Leon-Garcia, "SAVI testbed architecture and federation," in *Future Access Enablers for Ubiquitous and Intelligent Infrastructures (FABULOUS)*, Sep. 2015, pp. 3–10.

[90] D. Cock, A. Ramdas, D. Schwyn, M. Giardino, A. Turowski, Z. He, N. Hossle, D. Korolija, M. Licciardello, K. Martsenko, R. Achermann, G. Alonso, and T. Roscoe, "Enzian: An open, general, CPU/FPGA platform for systems software research," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, Feb. 2022, pp. 434–451.

[91] Z. István, G. Alonso, and A. Singla, "Providing multi-tenant services with FPGAs: Case study on a key-value store," in *Proceedings of the 28th International Conference on Field-Programmable Logic and Applications*, Sep. 2018, pp. 119–124.

[92] J. M. Mbongue, A. Shiuping, P. Bhowmik, and C. Bobda, "Architecture support for FPGA multi-tenancy in the cloud," in *International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Jul. 2020, pp. 125–132.

[93] J. M. Mbongue and C. Bobda, "Accommodating multi-tenant FPGAs in the cloud," in *Proceedings of the 28th IEEE Symposium on Field-Programmable Custom Computing Machines*, May 2020, pp. 214–214.

[94] J. M. Mbongue, D. T. Kwadjo, A. Shuping, and C. Bobda, "Deploying multi-tenant FPGAs within linux-based cloud infrastructure," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 15, no. 2, pp. 1–31, Jun. 2022.

[95] M. Gobulukoglu, C. Drewes, B. Hunter, D. Richmond, and R. Kastner, "Classifying computations on multi-tenant FPGAs," in *Proceedings of the 29th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2021, pp. 227–227.

[96] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: Toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, Oct. 2012.

[97] D. Bernstein, "Containers and cloud: From LXC to Docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[98] A. Vaishnav, K. D. Pham, and D. Koch, "A survey on FPGA virtualization," in *Proceedings of the 28th International Conference on Field-Programmable Logic and Applications*, Aug. 2018, pp. 131–138.

[99] *Intel Virtualization Technology (Intel VT)*, Intel, 2023. [Online]. Available: https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html

[100] C. Bobda, M. Majer, A. Ahmadinia, T. Haller, A. Linarth, and J. Teich, "The Erlangen slot machine: Increasing flexibility in FPGA-based reconfigurable platforms," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2015, pp. 37–42.

[101] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf, "Enabling FPGAs in hyperscale data centers," in *IEEE 12th Intl. Conf. on Ubiquitous Intelligence and Computing and 12th Intl. Conf. on Autonomic and Trusted Computing and 15th Intl. Conf. on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, Aug. 2015, pp. 1078–1086.

[102] O. Knodel, P. Lehmann, and R. G. Spallek, "RC3E: Reconfigurable accelerators in data centres and their provision by adapted service models," in *9th International Conference on Cloud Computing (CLOUD)*, Jun. 2016, pp. 19–26.

[103] F. Chen, Y. Shan, Y. Zhang, Y. Wang, H. Franke, X. Chang, and K. Wang, "Enabling FPGAs in the cloud," in *11th ACM Conference on Computing Frontiers*, May 2014, pp. 1–10.

[104] J. Mbongue, F. Hategekimana, D. T. Kwadjo, D. Andrews, and C. Bobda, "FPGAVirt: A novel virtualization framework for FPGAs in the cloud," in *IEEE 11th International Conference on Cloud Computing (CLOUD)*, Jul. 2018, pp. 862–865.

[105] S. Byma, J. G. Steffan, H. Bannazadeh, A. L. Garcia, , and P. Chow, "FPGAs in the cloud: Booting virtualized hardware accelerators with OpenStack," in *Proceedings of the 22nd IEEE Symposium on Field-Programmable Custom Computing Machines*, May 2014, pp. 109–116.

[106] S. A. Fahmy, K. Vipin, and S. Shreejith, "Virtualized FPGA accelerators for efficient cloud computing," in *International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov. 2015, pp. 430–435.

[107] J. Weerasinghe, R. Polig, F. Abel, and C. Hagleitner, "Network-attached FPGAs for data center applications," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2016, pp. 1–8.

[108] M. Asiatici, N. George, K. Vipin, S. A. Fahmy, and P. Ienne, "Virtualized execution runtime for FPGA accelerators in the cloud," *IEEE Access*, vol. 5, pp. 1900–1910, Feb. 2017.

[109] M. Vesper, D. Koch, and K. Pham, "PCIeHLS: An OpenCL HLS framework," in *International Workshop on FPGAs for Software Programmers (FSP)*, Sep. 2017, pp. 10–15.

[110] N. Tarafdar, T. Lin, E. Fukuda, H. Bannazadeh, A. Leon-Garcia, and P. Chow, "Enabling flexible network FPGA clusters in a heterogeneous cloud data center," in *Proceedings of the 25th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2017, pp. 237–246.

[111] J. Zhang, Y. Xiong, N. Xu, R. Shu, B. Li, P. Cheng, G. Chen, and T. Moscibroda, "The Feniks FPGA operating system for cloud computing," in *Asia-Pacific Workshop on Systems (APSys)*, Sep. 2017, pp. 1–7.

[112] J. Coole and G. Stitt, "Intermediate fabrics: Virtual architectures for circuit portability and fast placement and routing," in *International Conference on Hardware/Software Codesign and System Synthesis (CODESS+ISSS)*, Oct. 2010, pp. 1–9.

[113] N. Kapre and J. Gray, "Hoplite: Building austere overlay NoC for FPGAs," in *Proceedings of the 25th International Conference on Field-Programmable Logic and Applications*, Sep. 2015, pp. 1–8.

[114] X. Li and D. L. Maskell, "Time-multiplexed FPGA overlay architectures: A survey," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 5, pp. 1–19, Sep. 2019.

[115] P. Maidee, A. Kaviani, and K. Zeng, "LinkBlaze: Efficient global data movement for FPGAs," in *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec. 2017, pp. 1–8.

[116] J. M. Mbongue, D. T. Kwadjo, and C. Bobda, "FLexiTASK: A flexible FPGA overlay for efficient multitasking," in *ACM/SIGDA Great Lakes Symposium on VLSI (GLSVLSI)*, May 2018, pp. 483–486.

[117] A. Khawaja, J. Landgraf, R. Prakash, M. Wei, E. Schkufza, and C. J. Rossbach, "Sharing, protection, and compatibility for reconfigurable fabric with AmorphOS," in *13th USENIX Symposium on Operating Systems Design and Implementation*, Oct. 2018, pp. 107–127.

[118] M. Paolino, S. Pinneterre, and D. Raho, "FPGA virtualization with accelerators overcommitment for network function virtualization," in *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec. 2017, pp. 1–6.

[119] S. Pinneterre, S. Chiotakis, M. Paolino, and D. Raho, "vFPGAmanager: A virtualization framework for orchestrated FPGA accelerator sharing in 5G cloud environments," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, Jun. 2018, pp. 1–5.

[120] D. V. Vu, O. Sander, T. Sandmann, S. Baehr, J. Heidelberger, and J. Becker, "Enabling partial reconfiguration for coprocessors in mixed criticality multicore systems using PCI express single-root I/O virtualization," in *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec. 2014, pp. 1–6.

[121] Y. Zha and J. Li, "Virtualizing FPGAs in the cloud," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2020, pp. 845–858.

[122] ——, "Hetero-ViTAL: A virtualization stack for heterogeneous FPGA clusters," in *International Symposium on Computer Architecture*, Jun. 2021, pp. 470–483.

[123] W. Wang, M. Bolic, and J. Parri, "pvFPGA: Accessing an FPGA-based hardware accelerator in a paravirtualized environment," in *International Conference on Hardware/Software Codesign and System Synthesis (CODESS+ISSS)*, Nov. 2013, pp. 1–9.

[124] H. Yu, A. M. Peters, A. Akshintala, and C. J. Rossback, "Automatic virtualization of accelerators," in *Workshop on Hot Topics in Operating Systems (HotOS)*, May 2019, pp. 58–65.

[125] Y. Zha and J. Li, "When application-specific ISA meets FPGAs: a multi-layer virtualization framework for heterogeneous cloud FPGAs," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, Apr. 2021, pp. 123–134.

[126] D. Korolija, T. Roscoe, and G. Alonso, "Do OS abstractions make sense on FPGAs?" in *14th USENIX Symposium on Operating Systems Design and Implementation*, Nov. 2020, pp. 991–1010.

[127] J. Ma, G. Zuo, K. Loughlin, X. Cheng, Y. Liu, A. M. Eneyew, Z. Qi, and B. Kasikci, "A hypervisor for shared-memory FPGA platforms," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2020, pp. 827–844.

[128] M. Happe, A. Traber, and A. Keller, "Preemptive hardware multitasking in ReconOS," in *International Symposium on Applied Reconfigurable Computing (ARC)*, Apr. 2015, pp. 79–90.

[129] T. Xia, J.-C. Prévotet, and F. Nouvel, "Hypervisor mechanisms to manage FPGA reconfigurable accelerators," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2016, pp. 1–9.

[130] R. Elnaggar, R. Karri, and K. Chakrabarty, "Multi-tenant FPGA-based reconfigurable systems: Attacks and defenses," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, Mar. 2019, pp. 7–12.

[131] L. Bossuet and E. M. Benhani, "Performing cache timing attacks from the reconfigurable part of a heterogeneous SoC—an experimental study," *Applied Sciences*, vol. 11, pp. 1–14, Jul. 2021.

[132] F. Yao, M. Doroslovacki, and G. Venkataramani, "Are coherence protocol states vulnerable to information leakage?" in *International Symposium on High Performance Computer Architecture (HPCA)*, Mar. 2018, pp. 168–179.

[133] F. Liu, Q. Ge, Y. Yarom, F. Mckeen, C. Rozas, G. Heiser, and R. B. Lee, "CATalyst: Defeating last-level cache side channel attacks in cloud computing," in *International Symposium on High Performance Computer Architecture (HPCA)*, Mar. 2016, pp. 406–418.

[134] Y. Zhu, Y. Cheng, H. Zhou, and Y. Lu., "Hermes attack: Steal DNN models with lossless inference accuracy," in *Usenix Security Symposium*, Aug. 2021, pp. 1–16.

[135] R. Paccagnella, L. Luo, and C. W. Fletcher, "Lord of the Ring(s): Side channel attacks on the CPU on-chip ring interconnect are practical," in *Usenix Security Symposium*, Aug. 2021, pp. 645–662.

[136] A. G. Yağlıkcı, M. Patel, J. S. Kim, R. Azizi, A. Olgun, L. Orosa, H. Hassan, J. Park, K. Kanellopoulos, T. Shahroodi, S. Ghose, and O. Mutlu, "BlockHammer: Preventing RowHammer at low cost by blacklisting rapidly-accessed DRAM rows," in *International Symposium on High Performance Computer Architecture (HPCA)*, Mar. 2021, pp. 345–358.

[137] S.-Y. Tsai, M. Payer, , and Y. Zhang, "Pythia: Remote oracles for the masses," in *Usenix Security Symposium*, Aug. 2019, pp. 693–710.

[138] F. Hategekimana, J. M. Mbongue, M. J. H. Pantho, and C. Bobda, "Secure hardware kernels execution in CPU+FPGA heterogeneous cloud," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2018, pp. 182–189.

[139] B. Hettwer, J. Petersen, S. Gehrer, H. Neumann, and T. Güneysu, "Securing cryptographic circuits by exploiting implementation diversity and partial reconfiguration on FPGAs," in *Proceedings of the Design,*

Automation and Test in Europe Conference and Exhibition, Mar. 2019, pp. 260–263.

[140] S. Yazdanshenas and V. Betz, “Improving confidentiality in virtualized FPGAs,” in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2018, pp. 258–261.

[141] W. Lee, Y. Wang, and M. Pedram, “Optimizing a reconfigurable power distribution network in a multicore platform,” *IEEE Transactions on Computer-Aided Design of Circuits and Systems (TCAD)*, vol. 34, no. 7, pp. 1110–1123, Jul. 2015.

[142] I. Ahmed, L. L. Shen, and V. Betz, “Optimizing FPGA logic circuitry for variable voltage supplies,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 890–903, Apr. 2020.

[143] Z. Ebrahimi, B. Khaleghi, and H. Asadi, “PEAF: A power-efficient architecture for SRAM-based FPGAs using reconfigurable hard logic design in dark silicon era,” *IEEE Transactions on Computers (TCOMPUTER)*, vol. 66, no. 6, pp. 982–995, Jun. 2017.

[144] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. Irwin, and T. Tuan, “Reducing leakage energy in FPGAs using region constrained placement,” in *Proceedings of the 12th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2004, pp. 51–58.

[145] A. A. M. Bsoul and S. J. E. Wilton, “An FPGA architecture supporting dynamically controlled power gating,” in *Proceedings of the IEEE International Conference on Field Programmable Technology*, Dec. 2010, pp. 1–8.

[146] *UltraScale Architecture System Monitor - UG580*, Xilinx Corporation, Sep. 2021.

[147] *Intel Stratix 10 Analog to Digital Converter User Guide*, Intel Corporation, 2019.

[148] *AFI Power*, AWS GitHub, 2020. [Online]. Available: https://github.com/aws/aws-fpga/blob/master/hdk/docs/afi_power.md

[149] K. Vipin and S. A. Fahmy, “FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications,” *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–39, Jul. 2019.

[150] O. A. Uzun and S. Köse, “Converter-gating: A power efficient and secure on-chip power delivery system,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 2, pp. 169–179, Jun. 2014.

[151] E. Laohavaleeson and C. Patel, “Current flattening circuit for DPA countermeasure,” in *International Symposium on Hardware-Oriented Security and Trust*, Jun. 2010, pp. 118–123.

[152] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, “A side channel attack countermeasure using system-on-chip power profile scrambling,” in *2011 IEEE 17th International On-Line Testing Symposium*, Jul. 2011, pp. 222–227.

[153] V. Telandro, E. Kussener, A. Malherbe, and H. Barthelemy, “On-chip voltage regulator protecting against power analysis attacks,” in *2006 49th IEEE International Midwest Symposium on Circuits and Systems*, Aug. 2006, pp. 507–511.

[154] A. Moradi, M. Kirschbaum, T. Eisenbarth, and C. Paar, “Masked dual-rail precharge logic encounters state-of-the-art power analysis methods,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 9, pp. 1578–1589, Sep. 2011.

[155] A. Wild, A. Moradi, and T. Güneysu, “Evaluating the duplication of dual-rail precharge logics on FPGAs,” in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, Apr. 2015, pp. 81–94.

[156] A. Wild, A. Moradi, and T. Guneysu, “GliFreD: Glitch-free duplication towards power-equalized circuits on FPGAs,” *IEEE Transactions on Computers (TCOMPUTER)*, vol. 67, no. 3, pp. 375–387, Mar. 2018.

[157] J. Krautter, D. Gnad, and M. Tahoori, “CPAmap: On the complexity of secure FPGA virtualization, multi-tenancy, and physical design,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, pp. 121–146, Jun. 2020.

[158] F. Turan and I. Verbauwhede, “Trust in FPGA accelerated cloud computing,” *ACM Computing Surveys*, vol. 53, no. 6, pp. 128:1–128:28, Feb. 2021.

[159] S. Drimer and M. G. Kuhn, “A protocol for secure remote updates of FPGA configurations,” in *Proceedings of the International Workshop on Applied Reconfigurable Computing*, Mar. 2009, pp. 50–61.

[160] K. Eguro and R. Venkatesan, “FPGAs for trusted cloud computing,” in *Proceedings of the 22nd International Conference on Field-Programmable Logic and Applications*, Aug. 2012, pp. 63–70.

[161] K. Kepa, F. Morgan, K. Kosciuszkiewicz, and T. Surmacz, “SeReCon: A secure dynamic partial reconfiguration controller,” in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, Apr. 2008, pp. 292–97.

[162] P. Maene, J. Götzfried, R. de Clercq, T. Müler, F. Freiling, and I. Verbauwhede, “Hardware-based trusted computing architectures for isolation and attestation,” *IEEE Transactions on Computers (TCOMPUTER)*, vol. 67, no. 3, pp. 361–374, Jan. 2017.

APPENDIX

List of Acronyms

AES	Advanced Encryption Standard
API	Application Programming Interface
AWS	Amazon Web Services, *a subsidiary of Amazon*
AXI	Advanced eXtensible Interface, Microcontroller Bus Architecture
BNN	Binary Neural Network
BRAM	Block RAM, *used for storing data inside an FPGA*
BTI	Bias Temperature Instability
CAD	Computer-Aided Design
CARRY8	Fast carry logic for a CLB *(8 bit)*
CGRA	Coarse-Grained Reconfigurable Array
CLB	Configurable Logic Block
CMOS	Complementary Metal-Oxide Semiconductor
CPA	Correlation Power Analysis
CPU	Central Processing Unit
CSP	Cloud Service Provider
DMA	Direct Memory Access
DNN	Deep Neural Network
DoS	Denial-of-Service
DPA	Differential Power Analysis
DRAM	Dynamic RAM
DRL	Dual Rail Logic
DRM	Digital Rights Management
DSP	Digital Signal Processing *(chip)*
FPGA	Field Programmable Gate Array
FPGAVirt	A Virtualization Framework for FPGAs in the Cloud
GPU	Graphics Processing Unit
HARP	Hardware Accelerator Research Program, *an Intel platform*
HDL	Hardware Description Language
HLS	High-Level Synthesis
I/O	Input/Output
IP	Intellectual Property
JTAG	Joint Test Action Group, *a standard for verifying and testing chips after manufacture*
LAB	Logic Array Block
LUT	Look-Up Table
MAC	Mandatory Access Control
MLP	Multilayer Perceptron
MMIO	Memory-Mapped I/O
MPSoC	Multiprocessor SoC
NIC	Network Interface Controller
NN	Neural Network
NoC	Network-on-Chip
PAX	Post-Layout Parasitics
PCB	Printed Circuit Board
PCIe	Peripheral Component Interconnect Express, *a high-speed serial computer expansion bus standard*
PDN	Power Delivery Network
PID	Proportional-Integral-Derivative, *a widely used control mechanism*
PLL	Phase-Locked Loop
PUF	Physically Unclonable Function
RAM	Random-Access Memory
RSA	Rivest–Shamir–Adleman, *A public-key cryptosystem*
RO	Ring Oscillator
RTL	Register-Transfer Level, *a design abstraction for synchronous digital circuits*
S-box	Substitution-box
SAVI	Smart Applications on Virtual Infrastructure
SCA	Side-Channel Attack
SLR	Super Logic Region
SoC	System-On-Chip
SPA	Simple Power Analysis
SQL	Structured Query Language, *a data-base query language*
TDC	Time-To-Digital Converter
TEE	Trusted Execution Environment
TOR	Top-Of-Rack, *a switch that connects in-rack switches to the rest of the data center*
TRNG	True Random Number Generator
TTP	Trusted Third Party
VHDL	Very High-Speed Integrated Circuit Hardware Description Language
VM	Virtual Machine
XOR	Exclusive or