SoK: Secure FPGA Multi-Tenancy in the Cloud: Challenges and Opportunities

Ghada Dessouky
Technische Universität Darmstadt
Darmstadt, Germany
ghada.dessouky@trust.tu-darmstadt.de

Ahmad-Reza Sadeghi
Technische Universität Darmstadt
Darmstadt, Germany
ahmad.sadeghi@trust.tu-darmstadt.de

Shaza Zeitouni
Technische Universität Darmstadt
Darmstadt, Germany
shaza.zeitouni@trust.tu-darmstadt.de

***Abstract*—Field Programmable Gate Arrays (FPGAs) are increasingly deployed in datacenters due to their inherent flexibility over ASICs or GPUs that makes them an ideal processing unit for emerging and dynamic area of deep learning and other techniques and algorithms that are rapidly evolving. To maximize their utilization in the cloud, researchers have proposed the spatial *multi-tenant* deployment model, where the FPGA fabric is simultaneously shared among mutually distrusting tenants. This is enabled by leveraging the partial reconfiguration capability of FPGAs.**

In this paper, we systematize the research work on multi-tenant FPGAs in cloud computing settings and highlight their adversary models, security guarantees, as well as their fundamental security and privacy related shortcomings. We further categorize existing research works that demonstrate a new class of remotely-exploitable physical attacks on multi-tenant FPGAs by malicious tenants sharing physical resources with the victims. Through investigating end-to-end multi-tenant FPGA deployment comprehensively, we reveal that these attacks represent only one dimension of the problem, while various open security and privacy challenges remain unaddressed. We conclude with our insights on future research challenges and open opportunities.

***Index Terms*—Cloud FPGA Security, FPGA Multi-tenancy, FPGA-based Acceleration, FPGA-based Trusted Computing**

1. Introduction

Field Programmable Gate Arrays (FPGAs) are integrated circuits that can be (re)programmed after fabrication, as opposed to Application-Specific Integrated Circuits (ASICs), in order to implement custom functionality in hardware. FPGAs provide more flexible computing fabric than their ASIC counterparts, yet higher throughput and more computing power than their software counterparts. They require lower energy consumption, and given their continuously increasing capacities, they have been perceived to bring the best of both hardware and software worlds. In their steady growth and penetration of different application domains, FPGAs have also made their way into hardware acceleration of machine learning applications among other compute-intensive services. More recently, FPGAs have been increasingly adopted in datacenters to accelerate cloud-based services, such as by Microsoft [1], [2]. Other enterprises, such as Amazon [3], are offering clients to rent FPGAs in the cloud which they can freely configure with their own logic.

Multi-tenant FPGAs. To maximize FPGA utilization and the return-on-investment in cloud computing, researchers have proposed to share a single FPGA fabric among multiple users [4]–[10] by leveraging the *partial reconfiguration* property of FPGAs. This key distinguishing feature of FPGAs allows to reconfigure the functionality of a part, or *region* of the FPGA while being deployed in-field (at runtime). The reconfiguration property of FPGAs, both normal and partial, has led to the notion of *virtualized* or *multi-tenant FPGAs*, which are used interchangeably to refer to FPGAs shared among several tenants in the cloud. While this distinction is often not clearly outlined, FPGA sharing/multiplexing/partitioning can occur either *temporally* or *spatially*. Temporal sharing is where the FPGA device fabric, or the accelerator configured thereon, is used by different tenants at different time slots. This is the more conventional FPGA sharing model typically deployed in industry solutions. More recently, however, researchers have been investigating spatial sharing, where multiple tenants' designs can be co-located on the same physical FPGA device simultaneously, while occupying different logically isolated FPGA regions. Although this approach is principally possible and would further boost utilization, such a setting is not yet deployed in cloud computing architectures. It introduces a new threat landscape which we systematically investigate in this work.

Remotely-Exploitable Physical Attacks. Recent academic works have demonstrated the feasibility of remotely-exploitable physical attacks in multi-tenant settings, with a particular focus on spatial multi-tenancy. Such attacks have been shown to compromise the availability (e.g., DoS attacks) [11], [12], or integrity (e.g., fault injection attacks) [13], [14], or confidentiality (e.g., side- or covert-channel attacks) [15]–[30] of the victim's logic on the FPGA. The root cause for most of these attacks is that both the victim and malicious tenants 1) have their logic, while logically isolated, still co-located on an underlying fabric that shares resources such as the power supply system and are thus not physically isolated, and 2) have the freedom to configure their allocated regions with any (malicious) hardware logic of their choice.

FPGA-Based Trusted Computing. Besides the focus on boosting performance and utilization of FPGA resources and the emerging attack surface, literature on FPGA-based accelerated computing covers other emerging industry trends such as datacenter disaggregation [31] and edge cloud computing [32]. Nevertheless, very little has been invested in the direction of FPGA-based trusted computing [33]–[35] that aims to protect clients'

Intellectual Property (IP) on cloud FPGAs. Remotely-exploitable physical attacks, while clearly a threat, are indeed not the only security challenge stemming from these deployment settings. Other fundamental questions remain: how is clients' IP protection assured? How can FPGAs and their toolchains enable a minimal Trusted Computing Base (TCB) in order to provide the required security guarantees for clients and cloud service providers in different deployment scenarios?

Contributions. In this work we aim to provide a comprehensive overview of the security concerns in multi-tenant cloud FPGAs and shed light on specific directions for future research to address these challenges. Our main contributions are as follows.

- Introduction to the state-of-the-art trends in multi-tenant FPGA-based cloud computing.
- Systematization of the various security requirements and challenges in FPGA-based cloud computing and potential approaches to address them.
- Systematization of the different attacks in FPGA-based cloud computing. We classify these attacks into two major categories. The first category is remotely-exploitable physical attacks (remote physical attacks for brevity) that stem from the configurable nature of FPGAs where clients can freely configure their (malicious) hardware circuits on the FPGA. We further classify these attacks depending on the deployment model assumed, either spatial or temporal multi-tenancy. The second category includes more classical attacks that do not require the configuration of malicious primitives on the FPGAs and have their counterparts in CPU- or GPU-based computing.
- Systematization of defenses based on the different security requirements for secure multi-tenant FPGA settings (spatial and temporal).
- Insights on the open challenges and opportunities in enabling FPGA-based *trusted computing* in the cloud.

2. Field Programmable Gate Arrays

FPGAs are integrated circuits that can be electrically programmed or *configured* by end users to implement different digital circuits. Compared to ASICs, FPGAs are cost-effective and have shorter time-to-market. Moreover, unlike ASICs, FPGAs can be reconfigured to overwrite or update an existing design. However, the flexible nature of FPGAs comes at additional costs in terms of area, power consumption and performance, which are mainly attributed to the programmable interconnect of FPGAs [36]. FPGAs consist of three major components: configurable logic elements, configurable interconnects and input/output (I/O) blocks, which provide off-chip connections. Configurable logic blocks are connected together and to I/O blocks through the programmable routing interconnects to form the desired functionality. Other components of FPGAs are memory blocks (BRAM) and digital signal processing blocks (DSPs) for high-performance DSP applications, as shown in Fig. 1b [36].

FPGAs are usually integrated on printed circuit boards (PCBs) Fig. 1a, such that a single board can have one or more FPGA chips along with other components such as external interfaces to a computing platform, e.g., PCIe, USB, etc. A single FPGA chip can pack one or multiple dies, by leveraging advances in 3D integration and packaging, as in recent FPGA chips, e.g., Intel Stratix 10 and Xilinx Virtex UltraScale+. We show this distinction, since academic proposals called for leveraging the different chips per PCB or different dies per chip in the multi-tenant FPGAs in the cloud.

Design flow refers to the steps required to convert an abstract circuit description, written in a hardware description language, such as Verilog or VHDL, into a functioning circuit on a target FPGA. In *synthesis*, an abstract form of circuit description (functional, behavioral or structural) is translated into a functionally-equivalent gate-level representation, *netlist*, using a suite of different optimizations and mapping algorithms. Next, in *place-and-route*, the generated netlist is mapped onto the target FPGA's resources (LUTs, FFs, BRAMs, DSP cores, etc). In this step, the routing resources to connect the allocated resources are defined while preserving timing constraints, e.g., operating frequency. The developer can influence the outcome of this step by adding placement constraints as well. For example, the developer can restrict the design to be allocated on a specific region in the FPGA, or can force the routing of connections through specific channels. In *bitstream generation*, the actual binary file that configures the target FPGA is generated. Examples of FPFA toolchains provided by FPGA vendors are Intel Quartus Prime Software Suite and Xilinx Vivado Design Suite.

Partial Reconfiguration enables dynamically reconfiguring a portion of the FPGA, while the rest of the FPGA logic continues to operate seamlessly [37]. An FPGA can be partitioned into a static region and one or more reconfigurable regions (RRs), such that a reconfigurable region can be configured with its own *partial bitstream* without affecting other regions. This has been one of the most significant features of FPGAs in cloud-centric applications, since it allows the cloud service provider to partition the FPGA into one or more RRs executing different functions, thus enabling new features or future updates with increased flexibility and ease. Figure Fig. 1a shows an FPGA configuration with different RRs that can communicate with the CPU or other peripherals via pre-defined interfaces implemented in the static region. As long as the partial bitstream is accommodated by the resources of the allocated RR, the rest of the FPGA does not get affected by the dynamic reconfiguration of that region logic.

3. FPGA Deployment in the Cloud

While early FPGAs were primarily deployed in applications such as telecommunications and ASIC prototyping, more recently, they are used for large-scale datacenters and cloud computing services as *Acceleration-as-a-Service (AaaS)* or *FPGA-as-a-Service (FaaS)*. Both models provide *acceleration* for a wide range of compute-intensive workloads such as machine learning, genomic data processing, and other scientific computations.

In **AaaS**, FPGAs are dedicated to accelerate specific tasks that are pre-defined by the cloud service provider (CSP), e.g., a web search and network encryption. In this model, clients, a.k.a tenants, cannot freely configure the

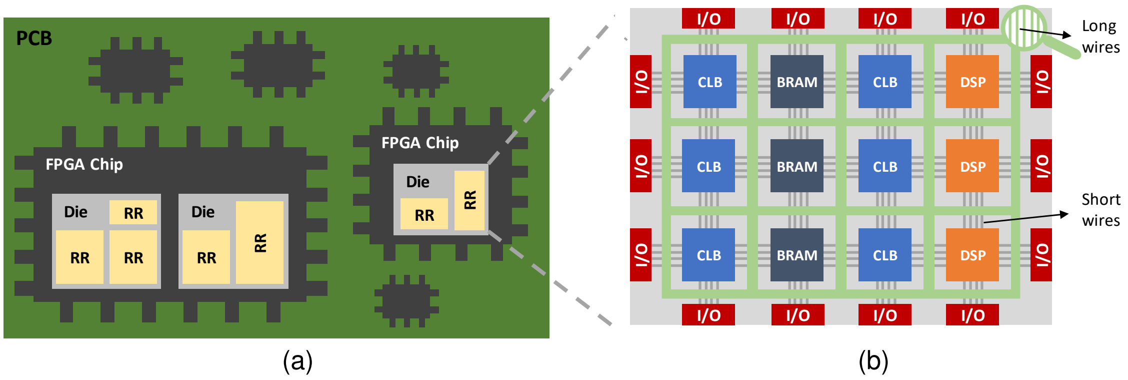


Figure 1. FPGAs on Board: (a) Printed Circuit Board (PCB), Chips & Dies with Reconfigurable Regions (RRs) & (b) Basic FPGA Structure [36].

FPGA fabric as desired, but can only benefit from the accelerated services.

FaaS, on the other hand, enables a more flexible usage where dedicated FPGAs are allocated to tenants and can be freely configured by them as desired. In FaaS, apart from acceleration, FPGAs have also been leveraged in some works [33]–[35] primarily to provide a hardware-based trust anchor for clients in the cloud under minimal trust assumptions. This deployment setting is often referred to as *FPGA-based trusted computing.*

Each of these different deployment settings impose different trust assumptions and requirements. For example, in *AaaS*, since FPGA devices are indiscernible/inaccessible to clients, only the FPGA vendor must be trusted, with no further trust assumptions or guarantees required. In *FaaS*, however, where the clients have more flexible access to the FPGA devices, additional security challenges arise, and thus trust assumptions and requirements differ. Furthermore, effective abstraction, virtualization and resource management techniques (briefly described in § 3.2) are also required.

In this work, we focus on the FaaS deployment model, and investigate the challenges and opportunities in achieving trusted computing in this setting under minimal trust assumptions. We present in the following section § 3.1 more details on the two FPGA-based acceleration deployment models, AaaS and FaaS, adopted by the cloud computing industry. Next, we present an overview on the different virtualization and resource management mechanisms proposed by academia for the FaaS model in § 3.2.

3.1. Commercial Deployment Solutions

FPGA-accelerated computing has been recently deployed by various commercial cloud services providers, e.g., Microsoft, and Amazon AWS. In the following we briefly look into CSPs and their deployment models.

AaaS. Microsoft Azure was among the first datacenters to introduce FPGAs in cloud computing by augmenting CPUs with an interconnected and configurable compute layer of FPGAs to accelerate portions of large-scale software services, e.g., Bing web search [38], or to reinforce the cloud infrastructure by performing network acceleration tasks [39], [40]. Microsoft demonstrated the first proof of concept that deployed FPGAs to accelerate web search ranking on its Bing web search engine [1]. Recently, Microsoft has announced its Project Brainwave [2], a deep learning acceleration platform, aiming to deliver real-time AI. The project aims to support the AI capabilities in Microsoft services like Bing web search and Skype language translations.

FaaS. Several CSPs offer their clients FPGA accelerated computing instances. Prominent examples are Amazon EC2 F1 [3], Huawei FACS FP1 [41], Alibaba F1 & F3 [42], Baidu FPGA Cloud Server [43], and Telekom ECS [44]. The differences among them are mainly in the allocated resources (RAM, vCPUs, network, etc.) and the available FPGA devices a client can rent. Depending on the selected instance capacity, multiple FPGA devices can be configured together to enable larger applications distributed efficiently across the FPGAs. Furthermore, the allocated FPGA devices are dedicated exclusively to the user during the paid period, i.e., spatial multi-tenant FPGA usage is not yet provided, as far as we can infer from the publicly available service description. These instances enable clients to develop FPGA-based services and accelerators by providing a Virtual Machine with pre-installed and licensed FPGA design tools , i.e., Intel Quartus, Xilinx Vivado or SDAccel, depending on the FPGA vendor as well as hardware and software development kits. These kits enable both expert hardware developers and non-specialist developers to generate the desired hardware logic. After the hardware development process is complete, the client uploads the resulting design netlist (the outcome of the synthesis step, cf. § 2) to the cloud. The corresponding FPGA image/bitstream is generated at the CSP side after inspection of the netlist against design rule checks. Then the client proceeds with the software development process, i.e., to develop, debug and run the applications that will benefit from the FPGA-accelerated tasks. Clients can develop their own FPGA-accelerated tasks for personal usage or for commercial purposes, where they can offer their accelerators in the corresponding cloud-based marketplace.

3.2. Academic Deployment Models

In this section we provide an overview of abstraction, virtualization and resource management techniques in the FaaS model as proposed in academic research.

In order to maximize resource utilization and return-on-investment of FPGA resources deployed in the cloud, these

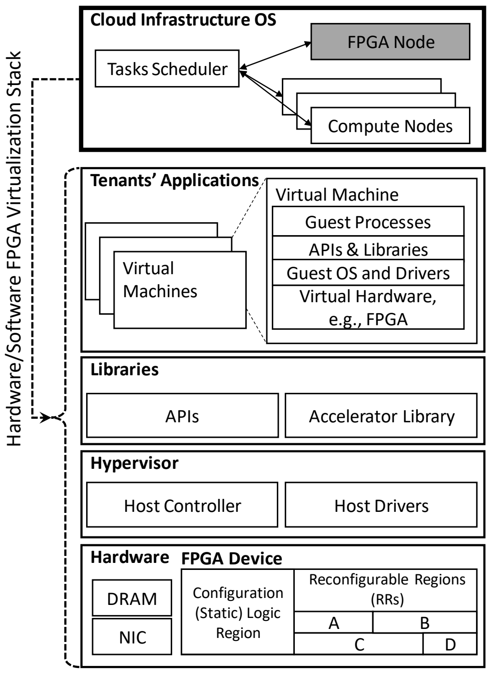


Figure 2. FPGA virtualization in typical cloud computing.

devices can be shared among different tenants, owing to the reconfigurable nature of FPGAs (cf. § 2). Virtualization is used to enable this sharing by abstracting away the complexities and intricacies of the underlying devices and simplifying the interface of interaction with the FPGAs. Earlier FPGA virtualization was primarily concerned with temporal FPGA multiplexing [45] to swap in and out partitions of larger designs when device capacity was limited. Over the years, FPGA virtualization has evolved significantly (in line with evolving FPGA capacities and capabilities), and is currently almost aligned with that of typical CPU and I/O virtualization [46], as shown in Fig. 2.

However, certain FPGA features, e.g., their heterogeneous fabric, execution model and how their logic configuration is vendor-dependent, differentiate them from typical CPU computing. Thus, this imposes specific challenges and requirements on how the virtualization can be handled, as we describe next.

Virtualization. There has been significant progress in FPGA virtualization in recent years [46]. Most of the state-of-the-art work, proposed by academia, focuses on performance boosting for the FaaS cloud-based model [4]–[10]. Nevertheless, both cloud-based models (FaaS & AaaS) require some mechanism of virtualization, similar to that in Fig. 2 to abstract hardware-specific details and represent the FPGA device and its components as a resource pool that can be actively managed, queried, allocated/de-allocated by tenants through the OS deployed in a cloud infrastructure. The cloud OS scheduler handles requests from tenants and creates virtual machines (VMs) for them, and schedules their tasks and resource requirements across a number of available resources and compute nodes, e.g., CPUs, memory, disks, as well as FPGAs. The tenants are thus provided access to their required resources via these VMs.

Granularity of Virtualization. The next question of FPGA virtualization concerns the level of granularity of FPGA primitives that is abstracted into a resource pool available to tenants. Abstracting FPGAs into registers, LUTs, I/O blocks, and memory blocks that can be managed by the cloud OS for tenants to request is, however, not trivial. This is because current FPGA-based development remains largely dependent on the specific hardware fabric of the FPGA and its layout details (cf. § 2). These details must be visible to the configuration toolchain provided in order to generate a compatible FPGA bitstream, where hardware-independent FPGA bitstream configuration remains an open research problem. Moreover, the irregular distribution of these different primitives across the FPGA fabric imposes spatial constraints on how the device gets partitioned into reconfigurable regions (RRs) and how user logic gets mapped to these regions. Recent FPGA devices, however, are attempting to achieve more regular distribution of primitives across the fabric [47].

Virtual FPGAs. Thus, FPGA virtualization schemes usually propose to provide a pre-defined pool of different fixed-size reconfigurable regions (RRs) of the FPGA [4]–[9], e.g. any of regions A - D in Fig. 2. Each reconfigurable region is considered a "virtual FPGA (vFPGA)" where any accelerator can be configured in it, as long as the region provides the required resources, thus providing some degree of independence from the FPGA fabric spatial specifics. Nevertheless, to enable higher flexibility with respect to the sizing of these RRs, recent work [10] has introduced a new layer of abstraction that encapsulates user FPGA logic and enables its mapping to dynamically-sized FPGA physical zones.

FPGA Shell. Besides RRs, some logic on the FPGA must remain statically configured (as shown in Fig. 2), sometimes referred to as the *FPGA shell* or hypervisor logic [6]. FPGA shell provides the physical infrastructure, i.e., clock signals, communication interfaces and reconfiguration management, that actually configures the remaining FPGA fabric (RRs A - D) with the partial bitstreams of the desired accelerator functionalities and controls their connections to FPGA interfaces.

Virtualization Stack. To share the available FPGAs and the RRs therein across multiple tenants as described, a *hypervisor* layer is required to provide host drivers to enable accessibility for the tenants to the underlying FPGAs. A controller module is also required to manage the resources allocation, perform address translation and provide bottom-level software interfaces to the FPGAs, thus providing multiple virtual instances of the FPGA devices [6]. *Libraries* are required to wrap up the available services (different FPGA configurations) into accelerator functions and maintain their bitstreams, while also providing respective APIs for tenants to call and use the accelerators from their software applications.

A tenant can use an API to call an accelerator function either i) from a *pre-defined pool* of different accelerator functions provided by the CSP such that the respective pre-compiled bitstream of the selected accelerator function is used to configure one of the available reconfigurable regions (RRs) [5], [6], or ii) of *own choice* i.e., a tenant may be allowed to provide an FPGA design that gets compiled into a compatible bitstream at the CSP side [5], [6], [9], or even be allowed to directly upload the accelerator's bitstream, given that the tenant is provided with the required information, i.e., vendor-dependent specifics and location on the FPGA to generate the partial bitstream [8].

The FPGA device itself can be either i) installed as a separate acceleration card that is either tied to a host CPU through PCI Express (PCIe) [6], [7], or configured as a stand-alone compute node that is accessed directly and independently over the network only [5], [8], [48] or over both the PCIe and the network [9], or ii) integrated into the same die package as the processor.

Memory accesses by the FPGA to the host's main memory use host physical addresses, whereas VMs on the cloud use guest physical addresses. Hence, an IOMMU is used to translate between the two address spaces to enable accelerators to access VM memory regions to read or write the required data. To further optimize the performance, FPGA memory accesses may be served from the host's last-level-cache (LLC) as well as an FPGA-exclusive local cache, which is kept coherent with the CPU's cache subsystem.[1]

FPGA (Temporal and Spatial) Multi-Tenancy. This virtualized deployment described above enables FPGA multi-tenancy, which traditionally, refers to *temporal* sharing of FPGA devices. Temporal sharing is where the FPGA device itself, or the accelerator functionality configured on it, is used by different users or tenants at different time slots (by some form of time multiplexing), but never simultaneously. This is the more popular FPGA sharing model and the one commonly deployed now in commercial solutions, as described earlier in § 3.1. More recently, however, there has been a growing interest in both academia and industry to enable *spatial* multi-tenancy on FPGA devices as well, where multiple tenants' logic are co-located on the same physical FPGA device simultaneously [49], [50]. In other words, mistrusting tenants would have their configuration bitstreams simultaneously occupying logically isolated RRs, e.g., A and B in Fig. 2. While such a deployment model is principally possible, and would ultimately enhance resource utilization and performance gains, it raises a multitude of challenges that have not been systematically investigated before. One of the key challenges that distinguishes this type of sharing on FPGA from CPU sharing across multiple tenants, is the security issues that arise, both at system- and physical-level, due to the sharing of the physical and raw hardware fabric/compute substrate itself, where the mutually mistrusting tenants have complete privilege and freedom to configure the underlying fabric (within their allocated RRs) as desired. Even if each tenant is restricted to configure the fabric of its allocated RR, the underlying hardware substrate and its physical implementation, e.g. the power supply and distribution are still shared, thus isolation at this level is challenging.

Performance vs. Security Gap. From a performance and efficiency standpoint, there is an abundance of significant works in the literature [4]–[10] that have been concentrated on boosting performance and resource utilization for FPGA-based accelerated computing, i.e., are *performance-centric*, as shown above, and assume a trustworthy CSP. On the other hand, from a security standpoint, the state of FPGA-based trusted computing lags behind. Very little has been invested in investigating and enabling a *security-centric* end-to-end deployment model that leverages cloud FPGAs to perform security-critical tasks on sensitive data, where clients can have their protected IP designs accelerated on cloud FPGAs without having to trust the CSP [33]–[35]. More details on the different adversary models of these different computing settings are presented next in § 4.

1. https://wiki.intel-research.net/FPGA.html

4. System Model & Threat Analysis

In this section we provide an analysis of the stakeholders, threats and resultant security requirements that emerge in FPGA cloud deployment settings.

4.1. Stakeholders

An overview of the of the system model of FPGA cloud computing deployment is shown in Fig. 3. We discuss next the involved stakeholders and their trust assumptions.

FPGA-based cloud services are commonly deployed by a *CSP*, which provides different usage models and services that involve heterogeneous architectures (cf. § 3). This usually involves a standard CPU-based host interfacing with co-processors and accelerator devices, such as GPUs and FPGAs, and offloading particular computation tasks to them as shown in Fig. 3. The CSP, being the platform owner of these facilities, may opt for temporal or even spatial multi-tenancy, i.e., offloading workload from different tenants to an accelerator device simultaneously, to maximize resource utilization and return on their investment, while reaping even higher performance gains.

FPGA vendors provide the FPGA devices and their toolchains to the CSP, which are in turn made available (besides additional development environments also provisioned by the CSP) for the tenants to use over the CSP provided service. FPGA devices and their toolchains are usually assumed trusted by the different stakeholders.

Clients or tenants rent the desired computation capacity and facilities and communicate their workload, which may include security-sensitive data as well as intellectual property (IP) design or code, to the CSP. Therefore, clients sharing resources in the cloud are mutually mistrusting. However, clients may trust the CSP with their sensitive data and only require that the CSP provides isolation among different tenants' workloads. Otherwise, clients may additionally also not trust the CSP, thus requiring it to deploy technologies such as Trusted Execution Environments (TEE) to isolate tenants' workloads from the system software and the CSP itself.

Besides tenants' IPs, the CSP, FPGA vendors and *IP partners* or enterprises, which are developing their own third-party IPs, may release their IPs in an IP marketplace (hosted by the CSP) for tenants to rent and use. To establish secure communication between the clients and the CSP, a *trusted authority (TA)* is required for the management and provisioning of digital certificates, which could either be another dedicated party or the CSP.

Further amplifying the complexity of these relationships, a very recent trend for maximizing datacenter efficiency and flexibility also involves the CSP disaggregating its facilities across multiple different distributed *resource pools* [31], and thus migrating the computation to these resource pools.

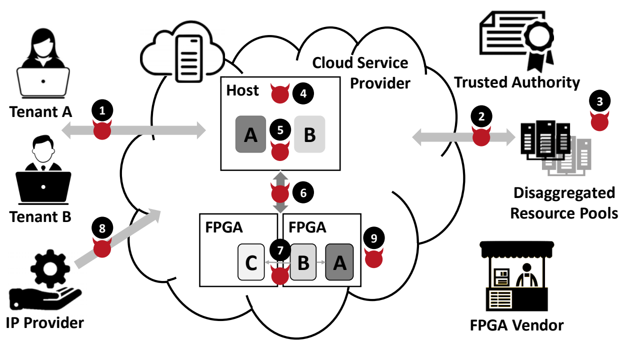


Figure 3. Stakeholders and their security concerns in FPGA-based cloud computing.

Other stakeholders are also involved, which we purposefully keep out of scope to keep the threat modeling more realistic and practical. Examples include foundry facilities that the hardware and FPGA vendors rely on for fabricating their devices, as well as independent entities that issue certificates for CSPs' compliance with security and privacy standards. We assume, however, that trusting these entities, especially fabrication facilities (for the CPUs and FPGA devices acquired by the CSP) is a reasonable and unavoidable assumption. Moreover, such threats and their proposed mitigation mechanisms occur at a very different layer of abstraction, and are an active and orthogonal research focus [51].

4.2. Threat Landscape & Security Requirements

Based on the system model and stakeholders described above, we identify two classes of security threats, demonstrated in Fig. 3. The first includes threats that are also common with traditional CPU-based cloud settings (§ 4.2.1) and do not require the configuration of malicious primitives on cloud FPGAs, while the second class of threats is exclusive to cloud FPGA devices (§ 4.2.2).

4.2.1. Traditional Threats. These are common to cloud-based VM settings and include:

Network Attacks. Tenant IP/data must be protected when transmitted to the cloud, whether a centralized cloud infrastructure ❶ or disaggregated/distributed cloud resources ❷, against an external network adversary.

Physical Attacks. Tenant IP/data must be protected at rest and in processing against an external adversary with physical access. Note that physical attacks are considered a threat only with disaggregation to the edge, where an external adversary may gain physical access to unattended edge devices, as opposed to a single, centralized and physically secure datacenter ❸.

Software Attacks. Tenant IP/data must be protected against potentially compromised hypervisor or system software ❹ and compromised co-tenants' workload on the host ❺. Examples of such attacks are software runtime attacks, microarchitectural and cache side-channel attacks, Rowhammer [52] and software-exploitable physical attacks [53]. We discuss in § 5 how some of these attacks can be also launched from FPGAs.

I/O and System Bus Attacks. Tenant IP/data, which are offloaded to accelerator devices, e.g., GPUs or FPGAs, must be protected when transmitted to/from the accelerator device ❻. This largely depends on, and may also influence, the choice of communication link between the host and the FPGA, e.g., physical link such as PCI Express, Ethernet or a wireless link. Possible attacks include man-in-the-middle attacks, payload misdirection, and device/communication spoofing or sniffing.

Countermeasures. Based on the cloud architecture and the assumed threat model some of these threats may not apply, e.g., physical attacks by the CSP are usually excluded and the CSP is assumed to protect the disaggregated cloud infrastructure against external adversaries with physical access. Other threats may not require explicit catering and assurances, if certain reasonable trust assumptions are instead made. For example, if tenants are willing to trust the CSP with their workloads, security concern ❹ is no longer valid, though standard cloud security measures should still be in place to isolate mistrusting co-tenants' workloads. However, if an untrusted cloud environment is assumed, clients may request to run their workloads in TEEs to isolate and protect co-tenants' workloads from the CSP itself [54]–[56].

4.2.2. FPGA-Specific Threats. This second class of threats stems from the reconfigurable nature of FPGAs and include:

Remote Physical Attacks based on Malicious FPGA Configurations. Tenant IP/data must be protected from untrusted and potentially malicious logic configured on the FPGA device. Damage, which can be caused by malicious logic executing on the FPGA, does not only affect co-tenants on the same physical FPGA but also other resources of the cloud infrastructure in general ❼. We discuss these attacks in detail in § 6 and § 7.

IP Theft or Corruption. Hardware IP providers must be protected from malicious access and counterfeiting, and be able to manage, query and monitor their IP licensing rights and usage ❽. This also includes protecting tenant IP by protecting the confidentiality and the integrity of FPGA bitstreams before and after configuration on the FPGA ❾.

Countermeasures. To tackle FPGA-specific threats, while assuming trusted FPGA devices and toolchains, additional trust requirements should be addressed to achieve secure spatial and temporal multi-tenant FPGA deployment in the cloud. We summarize their potential defenses below:

Defenses Against Remote Physical Attacks. These include i) preventive defenses, e.g., bitstream validation, that mitigate remote physical attacks *before* configuration on cloud FPGAs by detecting logic primitives that cause them (cf. § 8.1), and ii) runtime defenses that address the mitigation gap, which cannot be comprehensively closed by preventive defenses (cf. § 8.2).

IP Protection. Protection of confidentiality and integrity of IP bitstreams as well as secure configuration of bitstreams on cloud FPGAs either via direct or remote interfaces (cf. § 8.3).

In what follows, we describe the aforementioned two classes of attacks in more detail. The first class of attacks common to other cloud computing settings that do not require the configuration of malicious primitives on cloud FPGAs are described in § 5. The second class of attacks, remote physical attacks in multi-tenant FPGAs, are described in § 6 and § 7. The key enabler of the latter attacks is that clients can freely configure their (malicious) hardware circuits on the FPGA. For a better understanding, we provide a brief description of the malicious reconfigurable primitives deployed in these attacks in Appendix § A. These primitives include delay-line sensors, ring-oscillator (RO) sensors and RO power viruses. Most of these attacks assume spatial multi-tenancy (§ 6), however, some of the attacks are also applicable in the temporal multi-tenancy model adopted today in commercial cloud infrastructures (§ 7). All remote physical attacks and defenses are also summarized in Table 1.

5. Traditional Attacks

In this section we briefly outline attacks that are not FPGA-specific, i.e., they do not require the configuration of malicious primitives on FPGA logic in the cloud. These attacks are, thus, applicable in both FPGA deployment models (temporal or spatial) and have their equivalents in CPU- and GPU-based computing. They usually exploit non-configurable hardware and microarchitectural features that already exist in both computing systems, e.g., shared caches and DRAM-based main memories. Therefore, we also argue that the established defenses to mitigate these attacks in CPU-based computing can also be applied for FPGA devices, and we mention them briefly.

Rowhammer Attacks. Weissman et al. [74] demonstrated a software-based fault injection attack that leverages the Rowhammer effect in DRAM memories. By excessively accessing specific rows in the DRAM memory bit flips in neighboring DRAM cells can be induced [52]. The attack is demonstrated using two different setups[2], Intel Arria 10 FPGA integrated into the CPU package (A prototype E5-2600v4) and Intel Arria 10 acceleration card. Compared to a Rowhammer attack conducted by the CPU, the FPGA can achieve faster memory accesses and thus more bit flips due to the fact that, unlike memory accesses from the FPGA, memory accesses from the CPU must be followed by flushing the cached data. To conduct the Rowhammer attack, the FPGA is configured to perform memory accesses for specific amount of time, the parameters (target memory addresses and number of times to be accessed) are provided by the CPU.
Mitigation. As a countermeasure against these Rowhammer attacks, in line with these proposed for CPU-based attacks, the authors propose to increase DRAM row refresh rate. Another solution would be monitoring and detecting excessive DMA commands by the FPGA shell [6].

Cache Attacks. The feasibility of cache attacks on both the integrated Arria 10 FPGA and the Arria 10 acceleration card have also been shown by Weissman et al. [74] since the CPU and FPGA share memory and the last-level-cache (LLC). More specifically, the Arria 10 acceleration card has its own local DRAM besides access to that of the CPU. When the card reads from the CPU's memory, the request is served from the CPU's LLC if possible, otherwise from the main memory. Thus, the card is able to influence the state of the LLC and place cache lines into it, but only by writing (not reading) to the corresponding memory addresses. The integrated Arria 10 FPGA has access to the CPU host memory, and has its own dedicated directly-mapped local cache that is kept coherent with the CPU's cache. Depending on the type of memory access, it is either served directly from the CPU's LLC and main memory or it is first checked in the FPGA local cache. Then on a miss, the request gets forwarded to the CPU's LLC and main memory. Owing to both shared cache and memory, different cache-based attacks can be mounted by either the FPGA or the CPU to target the CPU's LLC or the FPGA local cache respectively. While eviction-based attacks can be mounted from the FPGA to influence and prime the CPU's LLC successfully, flush-based attacks (that require the `clflush` instruction) are not possible since the FPGA cannot run this instruction. On the other hand, eviction-based attacks that require to prime the FPGA local cache from the CPU are not possible, though flush-based attacks that leverage the `clflush` instruction are possible since the FPGA local cache is kept coherent with the CPU's cache. Furthermore, a covert channel can be constructed from the FPGA logic to the CPU by manipulation and probing of the LLC state.

Similarly, cache attacks in spatial multi-tenant FPGAs, where both the victim and attacker are co-located on the same FPGA device, would also be easily feasible. These attacks would exploit the FPGA local cache, so long as it is unconditionally shared among the different tenants, similar to how CPU cache is shared among different processes or cores. If the FPGA local cache is a directly-mapped cache, e.g. as in the Arria 10 FPGA, eviction set construction becomes trivial, especially since FPGA logic is usually allocated memory pages such that the requested memory buffer is within contiguous physical memory.
Mitigation. Cache partitioning mechanisms proposed to protect CPUs against these attacks, either software-based [75]–[77] or hardware-based [78]–[82], would, in principle, also mitigate cache attacks on FPGAs. However, to mitigate cache attacks in spatial multi-tenant FPGAs, where the CPU host OS is not involved, hardware-based defenses are the only feasible approach. Cache partitions would be utilized exclusively by the co-tenants, thus enforcing cache isolation among them and blocking side channels. Alternatively, specific cache lines that are security-critical can be tagged and locked in cache [78], [80] by hardware-based mechanisms, and thus cannot be evicted by other co-tenants. Hardware-based randomization defenses [83]–[85] proposed for cache attacks in CPU computing would also mitigate them with similar guarantees on FPGA devices.

6. Spatial Multi-Tenancy: Remote Physical Attacks

In spatial multi-tenant deployment model, a tenant that has the freedom to configure malicious primitives (cf. § A) on one of the virtual FPGAs, i.e., reconfigurable regions, can launch a physical attack on a victim logic occupying a virtual FPGA on the same physical FPGA as the attacker. For a better understanding of the attacks, we provide a brief description of the power supply system of an FPGA device next and classify the physical attacks based on the exploited physical phenomena into power leakage attacks § 6.1 and crosstalk effect attacks § 6.2.

Power Distribution Network (PDN) for an FPGA consists of a voltage regulator, decoupling capacitors and

2. https://wiki.intel-research.net/FPGA.html

TABLE 1. Remote Attacks and Defenses at a glance.

	Leakage Source			
Remote Physical Attacks	Power Leakage	Power Drop	Crosstalk	Thermal Leakage
Spatial Multi-Tenancy	[15]–[19], [21], [22]	[13], [14]	[23]–[27]	-
Temporal Multi-Tenancy	[28], [29]	[11], [12]	-	[30]
Defenses				
Bitstream Validation	[57], [58]			
Other Preventive Solutions			[25], [26], [59]–[61]	
Runtime Solutions	[20], [62]–[70]	[71], [72]		[30], [72], [73]

a grid of interconnects, modeled as an RLC-network, to deliver the power and ground voltages to FPGA components [86], [87]. The voltage drop V_{drop} is modeled as the sum of two components a *steady-state* voltage drop IR, caused by the interconnects' resistance of the RLC-network, and a *transient* voltage drop Ldi/dt, caused by the interconnects' inductance of the RLC-network. The power consumption is a product of voltage drop and drawn current. Since the voltage supplied to the FPGA device must remain fixed at a pre-defined level, changes in power demands, which vary based on the running functionality, are manifested as changes in current demands. The voltage regulator and the decoupling capacitors operate in parallel to accommodate the changes in current demand and to compensate steady-state and transient voltage drops, respectively, to try and maintain a fixed voltage level. However, in case of sharp transient changes in current demand of the FPGA, neither the voltage regulator nor the decoupling capacitors can respond fast enough. This leads to spatial and temporal fluctuations of the voltage level in the PDN, which may result in functional failures.

6.1. Power Leakage & Power Drop Attacks

Power consumption of victim cryptographic cores, configurable or running as a software (SW) process on an FPGA-based SoC, can be measured using either one of the configurable voltage sensors to retrieve their secret keys. Further, power leakage can be leveraged to construct a covert channel between malicious co-tenants. In the following we categorize state-of-the art attacks into side and covert channels.

Side-Channel Attacks. Schellenberg et al. [15] deployed a delay-line sensor to detect voltage fluctuations in the PDN caused by the activity of a soft AES-128 core, both located on the same FPGA fabric and logically isolated by passive fence of unused logic [88]. Using correlation power analysis (CPA), encrypted messages and the measured sensor values, i.e., power traces, the secret key can be recovered. The attack, which is demonstrated locally (in controlled Lab settings) on a SAKURA-G board with Xilinx Spartan 6 FPGAs, is shown to be successful for different locations of the sensor with reference to that of the AES core, but fewer traces are required when the sensor is closer to the AES core. The same attack was conducted on AES using several RO-based sensors in [16].Further, the attack is deemed feasible on Xilinx UltraScale+ FPGAs deployed in Amazon F1 instances. In [17] the AES key was fully recovered in 42% of 30 attempts made on different Amazon F1 instances.

In [18], RO-based voltage sensors are leveraged to monitor the activity of a soft RSA-1024 core, based on square-and-multiply, co-located on the same FPGA fabric. Using simple power analysis (SPA) and the sum of power traces from 20 RO-based sensors, RSA private key can be fully recovered. We summarize the details in Table 2. Further, the authors demonstrated that the start of RSA operations cannot be identified from the power traces when a noise generator, whose area overhead and power consumption are comparable to that of the RSA core, is activated. The attack also works when the RSA encryption runs as a process on a CPU that shares the same PDN with the FPGA, i.e., in an FPGA-based SoC. The attacks are demonstrated locally on a Zedboard deploying Zynq-7020 SoC of ARM Cortex-A9 and a Xilinx Artix 7 FPGA.

Similarly, Gravellier et al. [19] demonstrated a CPA attack on AES-128, running as a bare metal program on the CPU of a Zynq-7000 SoC, using delay-line sensors, implemented on the Artix 7 FPGA. In Table 2 we summarize the details of the aforementioned attacks.

Krautter et al. [20] investigated the effect of different placement strategies (default, area-, performance- or power-optimized) as well as the effect of process variation (PV), within a single FPGA instance (intra-PV) and among different FPGA instances of the same family (inter-PV), on the number of required traces by a CPA attack. The authors further showed a difference in the minimum amount of traces required for a successful attack of more than 100*x* for different locations of the AES core and the sensor on a Virtex 7 ADM-PCIE-7V3 accelerator card, which stresses the great impact of process variation on this attack.

Covert-Channel Attacks. By leveraging groups of RO-based power viruses as transmitters and RO-based voltage sensors as receivers, Giechaskiel et al. [21] demonstrated a one-direction covert channel between two users spatially sharing a cloud FPGA. Although spatial multi-tenant FPGAs are not yet adopted in real-world cloud solutions, such covert channels are shown feasible on Amazon F1 and Huawei FP1 instances, which deploy Xilinx *3D* Viretx UltraScale+ FPGAs. A Virtex UltraScale+ FPGA can accommodate up to four dies called super logic regions (SLRs) in a single chip to deliver high logic density [89], where the SLR dies are aligned in one dimension and share power and clock delivery systems. The authors envisioned, in a spatial multi-tenant scenario, that each SLR would be dedicated to a different tenant and demonstrated the construction of a covert channel between two SLRs based on power leakage, which is confirmed to be stronger when the receiver and the transmitter occupy neighboring SLRs. The attack is demonstrated on local and cloud Virtex UltraScale+ FPGAs (Huawei FP1 and Amazon F1 instances). In Table 3 we summarize the required resources and the performance (accuracy and bandwidth) of the attack. Different bandwidth numbers (bits per second)

TABLE 2. Spatial multi-tenancy: Remote side-channel attacks leveraging power leakage.

Ref.	Crypto Core			Sensor Properties		Required	local	Analysis
	HW/SW	Data Path	Freq.	Type	Sampling Freq.	Encryptions	/cloud	
[19]	bare metal SW process	AES 32-bit	667 MHz	8 delay-lines (5-bit TDC)	200 MHz	87K	local boards	CPA
		AES 8-bit				111K		
	HW	AES 128-bit	10 MHz			4.5K		
[15]		AES 32-bit	24 MHz	1 delay-line (6-bit TDC)	24 MHz [a]	5K [b]		
[17]		AES 128-bit	8 MHz	1 delay-line (8-bit TDC)	96 MHz	500K	Amazon F1	
[16]		AES 128-bit	50 MHz	64 ROs & 8-bit counters	250 MHz	8K	local boards	
[18]		RSA 1024-bit	20 MHz	20 1-stage ROs & 14-bit counters	0.1 MHz	9		SPA

[a] Results apply also to other sampling frequencies: 48, 72 & 96 MHz.
[b] The results correspond to the key bit showing the highest correlation.

are due to the different operating frequencies: 300 MHz, 200 MHz and 125 MHz on local VCU118, Huawei FP1 and Amazon F1. Transmitter ROs, which are grouped into subsets and placed in different clock regions, are enabled and disabled simultaneously. The transmission between the transmitter and the receiver is based on the Manchester encoding scheme, in which a single bit of information is represented as a tuple: '1' as (1,0) and '0' as (0,1).

Similarly, Gnad et al. [22] demonstrated a power covert channel, however, using a different encoding scheme. To send a '1' the transmitter generates positive voltage spike in the PDN by gradually activating groups of the RO-based power viruses and disabling all of them at once. Whereas to transmit a '0' a negative spike in the power trace should be generated by activating all power viruses at once and gradually disabling them. The receiver is a delay-line sensor that can measure the emerging voltage spikes in the PDN. 25 clock cycles separate between the transmission of two bits to allow the PDN to recover. The covert channel is evaluated on different boards: Pynq-Z1 with Zynq-7020 SoC (Xilinx Artix 7 FPGA) and KC705 with Kintex 7 FPGA. We present in Table 3 the best performance results when the transmitter and the receiver operates at 200 MHz and no additional noise source is running on the Kintex 7 FPGA.

Fault Injection Attacks. Krautter et al. [13] showed that by precisely controlling a power-hungry circuit, which comprises thousands of RO-based power viruses and occupys 30-50% of FPGA LUTs of a Terasic DE1-SoC board deploying Intel Cyclone V, it is feasible to induce timing failures in the 9th round of a soft AES-128 core, i.e., implemented in the FPGA fabric. The induced timing faults are used to recover the secret key by means of a differential fault analysis (DFA) [90], where the victim AES core operating at 111 MHz and spatially shared the FPGA with an array of RO-based power viruses. The achieved key recovery rate is of 93% on average using 18K encrypted messages. Mahmoud et al. [14] also leveraged a power-hungry circuit, occupying 25% of FPGA LUTs of a VC707 board with Xilinx Virtex 7, to induce timing failures during the operation of a true random number generator that are used for the generation of secret keys. The faulty output was shown to be biased and failed randomness tests.

6.2. Crosstalk Attacks

The crosstalk effect is caused by the electromagnetic coupling that occurs between unshielded wires that are laid in parallel. Thus, a signal transmitted through one wire influences the signal propagation delay in neighboring wires. The crosstalk effect is leveraged to launch remote side- and covert-channel attacks.

Side-Channel Attacks. Ramesh et al. [23] exploited the crosstalk effect within long wires in FPGA (cf. Fig. 1b) to recover the secret key of a soft AES-128 core. The transmitter, routed through long wires, is an input to an AES S-box. The snooping circuit comprises a RO-based voltage sensor. The receiver, which is one of the RO's wires, is routed through long wires and is located next to the transmitter wire. Using differential power analysis (DPA) [91], the counter values obtained from the snooping circuit (power traces) and the encrypted messages, the key of the final round of the AES can be recovered byte by byte. Then the encryption key can be obtained by inverting the key schedule. The crosstalk effect increases for longer transmission pair and lower operating frequencies of the victim logic, which directly influences the number of encryption operations required to recover the key. The attack is evaluated locally on several boards of Intel FPGA devices: DE2-115 board, Cyclone IV and Stratix V development kits [23] and DE5a-Net Arria 10 GX boards [24].

The same effect is also demonstrated in [25], where the authors leverage a sliding-window sampling scheme to sample the RO-based sensor at overlapping time periods in order to extract the value of each transmitted bit of a secret key. The authors show that the probability of correctly recovering a key is higher for larger keys (asymmetric vs. symmetric keys) and smaller window sizes. In Table 4 we summarize the performance of each of the aforementioned DPA and sliding window approaches. Long wire leakage is characterized for different generations and families of Xilinx FPGAs, including Virtex 5, Virtex 6, Artix 7, Spartan 7 and Virtex UltraScale+, on local boards [26] as well as on cloud FPGA instances [27]. More technical details on the respective attacks can be found in Appendix § B.

Covert-Channel Attacks. Besides side-channel attacks, the crosstalk effect can be exploited to construct a covert channel between two circuits sharing the FPGA fabric at the same time [25], [26]. Crosstalk-based covert channels have been shown locally on several boards (of Xilinx FPGA devices) in [26]: ML509 (Virtex 5), ML605 (Virtex 6), ArtyS7 (Spartan 7), Nexys 4 DDR and Basys 3 (Artix 7). In Table 3 we summarize the performance of a crosstalk covert channel conducted with transmission pair of 2-segment long wire and Manchester encoding

TABLE 3. SPATIAL MULTI-TENANCY: REMOTE COVERT-CHANNEL ATTACKS.

Ref.	Deployed Effect	Bandwidth	Error Rate	Platform	Transmitter	Receiver
[21]	Power Leakage	1.17 Mbps	0.1%	local VCU118	4K 2-stage ROs (Fig. 4c)	25 7-bit counters & 2-stage ROs (Fig. 4c)
[21]	Power Leakage	781 Kbps	0.1%	Huawei FP1	6K 2-stage ROs (Fig. 4c)	
[21]	Power Leakage	488 Kbps	0.1%	Amazon F1		
[22]	Power Leakage	8 Mbps	0	local KC705	5K 1-stage ROs (Fig. 4c)	6-bit delay-line
[26]	Crosstalk	6.1 Kbps	0.1 - 1%	local boards	one signal buffer	single 21-bit counter & 3-stage RO (Fig. 4b)

TABLE 4. SPATIAL MULTI-TENANCY: REMOTE SIDE-CHANNEL ATTACKS LEVERAGING CROSSTALK EFFECT.

	Victim Circuit			Snooping Circuit	Required	
Ref.	Key Size	Operating Freq.	Transmitter Length	Type / Sampling Freq.	Encryptions	Analysis
[23]	AES (128-bit)	1 MHz	1-segment long wire	RO / 1 MHz	233K [a]	DPA
[26]	32-bit	0.78 MHz	$\geq$ 1-segment long wire	RO / 195 KHz	20K [b]	Sliding Window ($w = 4$)

[a] Reported results are for automatically placed and routed victim AES.
[b] For all four counters, i.e., 5K per counter, to recover only 98.4% of the 32-bit key.

scheme without error-correcting codes. Compared to a covert channel based on power leakage, crosstalk effect has notably lower performance (accuracy and bandwidth), however, it also requires less FPGA resources.

To summarize, exploiting the crosstalk effect requires more measurements or traces than a power side-channel attack, as seen in Table 2 and Table 4. Further, exploiting crosstalk effect requires close proximity between the victim and snooping circuit with a maximum distance of one long wire separating the transmitter and the receiver, which is hard to achieve in a real-world deployment model, given that a malicious tenant has no influence on the process of place-and-route of the victim netlist on the FPGA. Moreover, It is demonstrated that the crosstalk effect diminishes for victim circuits operating at higher frequencies [23]. In general these attacks, whether based on power leakage or crosstalk effect, are demonstrated on cryptographic cores operating at low frequencies, e.g., 24 MHz for the AES core, as opposed to the purpose of leveraging FPGAs to achieve the high performance these platforms can offer.[3] Running the cryptographic cores at higher frequencies of few hundreds MHz would require higher sampling rates that are harder to achieve with reconfigurable sensors.

7. Temporal Multi-Tenancy: Remote Physical Attacks

In this section, we present remote physical attacks that assume the conventional temporal multi-tenant FPGA deployment model, which is adopted in real-world cloud infrastructures. That is the whole physical FPGA is allocated to one client at a time. Nevertheless, different tenants can use different FPGA instances (and other resources) at the same time. We classify these attacks into power leakage attacks § 7.1 and thermal leakage attacks § 7.2.

7.1. Power Leakage & Power Drop Attacks

Side-Channel Attacks. Schellenberg et al. [28] deployed a delay-line sensor in one FPGA to measure the power consumption of a soft cryptographic core (AES-128 or RSA-224) implemented on another FPGA that is integrated on the same board and sharing the power supply system. To enable the comparison of results with spatial power side-channel attacks (cf. § 6.1), the authors replicated their settings from [15] (same board and same AES core). They showed that to recover AES key, 2.5 million encryptions are required when the AES core and sensor are on two different FPGAs, compared to 5K encryptions when they are co-located on the same physical FPGA. The attacks are demonstrated on a SAKURA-G board integrating two Xilinx Spartan 6 FPGAs.

Covert-Channel Attacks. Power leakage is further exploited on three realistic setups of FPGA-to-FPGA, CPU-to-FPGA and GPU-to-FPGA covert channels, where transmitters and receivers are separated on two distinct boards sharing only the external power supply unit in [29]. Nevertheless, to capture the power leakage across boards, additional set of RO-based power viruses, the stressor, is required. The stressor power viruses and the receiver sensors, which are on the same FPGA, are grouped into subsets and distributed across different clock regions in the corners and the center of the receiver FPGA, while the transmitter is different based on the source type, i.e., FPGA or CPU/GPU. In case of FPGA-to-FPGA the transmitter is a set of RO-based power viruses, whereas in case of CPU-to-FPGA and GPU-to-FPGA the transmitter is a stress program that runs simultaneously on available threads or cores. During the transmission of one bit, the stressors are alternately enabled and disabled in equal intervals and the counter values of each receiver RO for the enabled and disabled periods are averaged and compared to extract the transmitted bit. Then the final transmitted bit is computed based on the majority of all receiver ROs. The covert channel is demonstrated locally on different FPGA boards KC705 (Xilinx Kintex 7) and AC701 (Xilinx Artix 7), two Xeon processors with 4 threads and 24 threads, and two GeForce GPUs with 96 and 640 CUDA cores. In Table 5 we summarize the best performance results (bandwidth and accuracy) and the required resources to achieve them.

By comparing the results in Table 3 and Table 5 in terms of resources and performance, it is obvious that power covert channels in spatial settings are stronger and require less resources.

3. For example, AES can run at 300MHz on a Virtex 6 FPGA https://opencores.org/projects/tiny_aes.

Denial of Service (DoS) Attacks. Gnad et al. [11] demonstrated that excessive switching activities of a power-hungry circuit, occupying 12-13% of the FPGA LUTs on Xilinx ML605 (Virtex 6), KC705 (Kintex 7) and Zedboard (Artix 7), in the picosecond regime create voltage emergencies in the PDN. Such sharp and transient voltage drops cannot be compensated by the decoupling capacitors or the voltage regulator and cause the FPGA to crash in less than 1 ms. The DoS attack is demonstrated on local boards and is deemed feasible on Xilinx UltraScale+ FPGAs similar to those deployed in Amazon EC2 F1 instances [12]. Note that the size of the power-hungry circuit and the required resources on the FPGA to induce the required effect differ based on the targeted FPGA.

7.2. Thermal Leakage Attacks

Tian et al. [30] demonstrated how to exploit thermal leakage of cloud FPGAs to establish a covert channel using single FPGA. This requires a pre-agreement between the clients on the identity of target FPGA instance, i.e. the communication medium, which implies disclosing information about the cloud infrastructure, i.e., identifying FPGA instances offered by the CSP. One approach to acquire such information is by fingerprinting cloud FPGAs using physically unclonable functions to measure unique characteristics of the FPGAs. Tian et al. [92] conducted the experiments on Amazon AWS cloud and demonstrated the feasibility of renting the same FPGA by two successive clients for the different available instances.

By leveraging a grid of ROs as a transmitter (heater) and RO-based thermal sensors as a receiver, Tian et al. [30] demonstrated a covert channel between two clients temporally sharing the same cloud FPGA. A transmitter circuit is configured on the FPGA and enabled to achieve a steady-state temperature corresponding to logic '1'. Then the transmitter circuit is disabled and the FPGA is left idle before configuring the receiver's bitstream on the same FPGA. The receiver circuit then measures the ROs frequencies and compares them to nominal ROs frequencies which correspond to logical '0' to acquire the transmitted value. The covert-channel bandwidth is defined by the heating period, the maximum time gap between heating the FPGA and sensing its temperature, and the transmitter's size (number of ROs). The longer the idle gap/period between heating and sensing, the weaker the covert channel is. The evaluation is performed on cloud FPGA instances, where for an idle period between two users of a cloud FPGA of 120 seconds, 6.64 minutes are required to transmit one bit. Unlike covert channels based on crosstalk or power leakage, thermal leakage impose no constraints on the placements of transmitter and receiver circuits since the generated heat can be sensed from anywhere on the FPGA. The covert channel is demonstrated on Stratix V FPGAs deployed in Texas Advanced Computing Center. Table 5 shows the performance of the thermal covert channel and a comparison to power covert channels.

8. Defenses

We classify state-of-the-art defenses against remote physical attacks (cf. § 6 and § 7) into preventive solutions that mitigate the threats before configuration on the FPGA (§ 8.1) and runtime solutions that aim to address the mitigation gap that static approaches cannot comprehensively close (§ 8.2). In § 8.3 we examine IP/data protection and secure configuration solutions.

8.1. Preventive Solutions

8.1.1. Bitstream Validation. This attempts to verify that a user design is harmless, i.e., contains no malicious primitives that can be deployed for remote physical attacks. One way to do this is by leveraging design rule checks (DRCs), which are provided in FPGA vendor toolchains, e.g., Xilinx Vivado DRCs, to verify that a design does not violate a set of pre-defined rules. Amazon AWS enforces DRCs on clients' netlists to prevent combinatorial loops, e.g. ROs, before generating the final bitstream. [4] However, alternative sequential RO designs (cf. § A.1) can still bypass the check [27], [93], [94]. In particular, latch-based and flip-flop based RO sensors are used to characterize long wire leakage on Amazon F1 and Huawei FP1 instances [27].

Another approach for bitstream validation is by using stand-alone virus scanner tools [57], [58] that search for uncommon structural and behavioral properties of malicious logic in the netlist. For example, FPGADefender [58] is designed to detect combinatorial and sequential self-oscillating circuits, whereas the virus scanner in [57] is designed to detect combinatorial loops as well as delay-line sensors. Such tools are envisioned to be deployed by the CSP to scan the uploaded FPGA circuits prior to configuration on the FPGA. In case the CSP allows clients to directly upload their bitstreams, reverse-engineering of the bitstream is required to recover the netlist. Moreover, the virus scanner tool must be updated regularly with formulation of properties, i.e., virus signatures, to account for new malicious logic designs.

Both DRCs and virus scanners require access to plain-text client IP designs, which introduces a conflict with IP confidentiality (discussed further in § 9.1).

8.1.2. Prevention of Crosstalk Attacks. Giechaskiel et al. [25], [26] proposed to block four neighboring long wires, two long wires from each side, of the victim wires. The blocked wires can be left unoccupied or further driven by random values. This can be performed either by manual inspection of the placed and routed design to identify the source of leakage and add the guarding wires or by the support of FPGA toolchains. This requires including directives in the source code of the design by the developer to annotate sensitive signals and further modifying routing algorithms to automate adding the guarding wires.

A compromise solution is proposed by Luo and Xu [60]. They implemented a framework in the command line interface of FPGA toolchains to prioritize the placement of sensitive logic based on their security-levels in a spiral manner such that the most sensitive logic is in the center of the allocated blocks. For example, the S-Boxes of the AES core are placed in the center of the allocated logic and are routed away from long wires. External interfaces to the sensitive logic, e.g., UART or PCIe, routed through long wires are guarded by driving the neighboring long

4. https://github.com/aws/aws-fpga/blob/master/ERRATA.md

TABLE 5. TEMPORAL MULTI-TENANCY: REMOTE COVERT-CHANNEL ATTACKS.
ALL ATTACKS USE 3-STAGE RO DESIGNS (RO FIG. 4A IN [30] AND RO FIG. 4B IN [29])

Ref.	Deployed Effect	Bandwidth	Error Rate	Platforms	Transmitter	Receiver	Stressor
[29]	Power	6.1 bps	$\leq 5\%$	KC705-to-AC701	14K ROs	20 ROs & 15-bit Counters	500 ROs
[29]	Power	6.1 bps	3%	CPU-to-AC701	14 threads		
[29]	Power	2 bps	3%	GPU-to-AC701	640 CUDA core		
[30]	Thermal	1 bit per 6.64 minute	0%	Altera Stratix V	41K ROs *	RO sensors *	N.A.

* Exact numbers are not given.

wires with '0', '1' or random values. The framework operates at the netlist level of the design, therefore, it is unlikely that additional resources (LUTs or memory blocks) will be incurred. Nevertheless, constrained place-and-route would affect the maximum operating frequency of the circuit, besides the two additional long wires, surrounding each sensitive long wire, that are reserved and driven by obfuscation logic.

Seifoori et al. [61] demonstrated that modifying the PathFinder FPGA routing algorithm eliminates the crosstalk effect in the context of integrating untrusted third-party IP cores and security-critical IP cores in a single design. The authors proposed four routing strategies that either i) entirely block the nearest two or four neighbors of a sensitive wire, ii) block only untrusted signals (of a different core) from occupying the two neighbors of a sensitive wire or iii) allow only trusted signals (of the same core) to occupy the two neighbors of a sensitive wire. The performance loss is evaluated for each strategy in terms of routing channel width and critical path delay (maximum operating frequency).

Xilinx Vivado toolchain includes an Isolation Design Flow, which enforces physical separation between different cores/designs on the FPGA, such that the entire design (functions & signals) is placed-and-routed inside the defined region [59]. Then, Vivado Isolation Verifier can be used to ensure that isolation rules are met. This mitigates the crosstalk leakage, however, it may introduce performance issues due to routing constraints.

In summary, mitigating crosstalk should be adopted by the CSP to secure both clients logic and their interfaces to the FPGA shell in case clients are only allowed to upload their designs/netlists. Otherwise, clients should make sure that their circuits are not routed through long wires, whereas the interfaces to the FPGA shell should be still secured by the CSP.

8.2. Runtime Defenses

We refer here to runtime solutions that aim to i) detect 'malicious' activities by detecting changes in operating conditions or ii) prevent or harden the exploitation of the leakage during the execution on the FPGA.

8.2.1. Detection of Power-Drop Attacks. Provelengios et al. [71] proposed to deploy a network of RO-based sensors in the FPGA shell to periodically detect and identify malicious voltage drops and to locate the center of a power-hungry circuit. This information is used by CSP to revoke the malicious logic from the FPGA. The accuracy of this approach depends highly on the number of sensors and also the intensity of the attack (number of power viruses). The more sensors and the intenser the attack, the more accurate the results will be. In their settings, the authors used 46 RO sensors (each comprises a 19-stage RO & a 20-bit counter) resulting is 5% resource utilization on a Intel Cyclone V FPGA of DE1-SoC board. Although the authors do not elaborate on the time required to locate the center of the malicious logic, it is very likely that until the malicious logic is located and revoked, timing errors in neighboring logic would still occur or the FPGA would be shut down. Alternatively, when voltage drops are detected, the clocks of virtual FPGAs can be paused until the CSP locates the source of the malicious activities. Note that the clients need to trust the CSP, otherwise, a malicious CSP can deploy a sensor network to spy on the clients.

Xilinx provides a security monitor IP core that can be instantiated in some of their FPGAs [72]. This IP core monitors supply voltage and temperature in addition to other functionalities and allows the user to decide among different configurable penalties. Such IP core can be instantiated by the CSP to detect voltage drop in the FPGA and zeroize (regions of) the configuration memory for example. However, it is not clear whether it can provide information about the source of the voltage drop.

8.2.2. Mitigation of Power Leakage Attacks. Krautter et al. [62] proposed a hiding countermeasure, through which side-channel leakage is reduced by implanting a fence of ROs that works as a noise generator circuit around the logic to be protected, thus reducing signal-to-noise-ratio (SNR). The fence has a size of roughly the size of of the design to be protected and its power consumption is slightly more than half of the AES power consumption. Krautter et al. [62] further proved that a fence activation strategy based on internal power measurements, using voltage sensors, is more effective than a random activation pattern. Their approach increased the number of required traces to recover the key from 1.8K, when no defense applied, to 300K for sensor-based activated fence. Nevertheless, physical locations of the fence and AES core highly affect the size of the fence and the required traces for a CPA attack. Krautter et al. [20] showed that by carefully choosing the AES core location on the targeted FPGA, the same protection level of the active fence approach can be achieved with zero overhead. However, this requires exhaustive per-device analysis to find the least vulnerable location per FPGA.

Note that implementing an active fence by a client would not be allowed if a bitstream validation process is applied by the CSP. On the other hand, if such a defense is to be applied by the CSP, then the CSP needs to orchestrate active fences for the virtual FPGAs, resulting in additional and non-trivial overhead in terms of power and FPGA resources. Other hiding techniques based on dual-rail precharge logic or duplication schemes [63], [64] can be implemented directly by the clients in their allocated virtual FPGAs. However, they still suffer information leakage due to imbalanced routing on FPGAs and result in duplicated area and power overhead.

Other countermeasures that can be implemented by the client against power leakage on FPGAs are randomization and masking. Randomization approaches work by randomly changing operating frequency (if allowed by the CSP) [66], [67], [70], execution of random dummy operations [66] or adding random delays to input signals [68]. Masking techniques for reconfigurable platforms are either inefficient with respect to area and performance [69], or still generate exploitable leakage [65]. Moreover, masking techniques should be designed for every cryptographic function individually and the desired security guarantees.

8.2.3. Mitigation of Thermal Leakage Attacks. Intel integrates diode temperature sensors in the configurable fabric of their FPGAs [73]. To mitigate thermal leakage, these sensors can be exclusively leveraged by the CSP to monitor the temperature across the device at runtime. Once the temperature reaches a pre-defined level, the CSP can apply a suitable cooling countermeasure.

In [30] the authors proposed enforcing a minimum idle period, in the order of minutes, for the FPGA between two successive allocations. This allows the FPGA to recover to a nominal temperature. Alternatively, they proposed to heat or cool the FPGA to a pre-defined temperature before it is reallocated. This mitigation needs to be considered for practical feasibility and against the CSP's drive for operational efficiency where idle cycles are lost income.

8.3. FPGA IP & Data Protection

We present in the following an overview of academic solutions that are focused on IP protection in an untrusted cloud environment for FPGA-based *trusted computing*. Then we refer to other schemes that try to address data isolation and access control for cloud FPGAs.

8.3.1. IP Protection. State-of-the-art solutions rely on an initial bitstream that configures a trust anchor on a cloud FPGA. The main goal of this trust anchor is to run a key exchange protocol with clients and decrypt client' IP bitstream prior to configuration on the FPGA. The trust anchor contains a cryptographic core for decryption of IP bitstreams using secret session keys, which are obtained through the key exchange protocol that deploys public key cryptography, e.g., RSA or elliptic core cryptography (ECC) [33], [35], [95]. In [33], the authors proposed to embed the private key directly with the RSA core in the initial bitstream, thus, the confidentiality of the initial bitstream must be protected and this is achieved, e.g., by leveraging the built-in AES engine in Xilinx FPGAs. For that, the authors propose a trusted party, e.g., the FPGA vendor, to program the AES secret key before deploying the FPGA in the cloud. In [95], the authors do not protect the initial bitstream but rather obfuscate the public key into a finite state machine (FSM) [96]. Such that the public key cannot be inferred by static analysis of the initial bitstream, i.e., the initial bitstream must be configured on the FPGA in order for the trust anchor to generate the public key. The security of the scheme relies on strong assumptions of time-bounded protocol responses to detect network attacks, e.g., man-in-the-middle. Moreover, the unprotected initial bitstream is vulnerable to bitstream manipulation attacks [97]. In [35], the authors embed a physically unclonable function (PUF) in the trust anchor to generate device-specific session keys. This implies protecting only the integrity of the initial bitstream as PUF outputs are generated on-the-fly. However, the authors instead assume that the FPGA vendor configures the trust anchor on the FPGA prior to deployment in the cloud and that the FPGA is constantly powered, even during shipping to the CSP, to maintain its configuration. Alternatively, the initial bitstream of the trust anchor can be protected and stored in off-chip memory as in [33]. Note that the trust anchor, which is designed and implemented by the trusted party, may also include the rest of the FPGA shell functionality (interfaces to host). However, this would impose restrictions on the design decisions of the CSP.

In [34] an initial bitstream is not required. The authors leverage the built-in PUF, ECC and AES cores on SmartFusion-2 FPGAs of Microsemi. The PUF is used to generate a pair of ECC keys, and the client uses the ECC public key to encrypt the AES secret key. The AES key is used to encrypt client bitstream. Partial reconfiguration of the FPGA fabric is not available in these FPGAs, therefore the client should design and implement the interfaces to host, which is inconvenient for the CSP. On the other hand, in an untrusted environment, the FPGA can be run in factory test mode to read out client's IP design.

In [33], [34] the public key is published via standard public key infrastructure held by the trusted party.

These schemes mainly assume temporal multi-tenant setting. Nevertheless, [33], [35] would theoretically work in the spatial setting, however, a malicious tenant can overwrite the configuration of another tenant, since the CSP has neither control over the bitstream generation nor access to plain-text bitstreams. Moreover, the CSP has no guarantees that the encrypted bitstream is harmless, i.e., free of power viruses or sensors. To sum up, for trusted computing on cloud FPGAs, FPGA vendor support is required for IP protection. At the same time, the CSP must be assured that encrypted clients bitstreams contain no harmful logic. We discuss this in more detail in § 9.1.

8.3.2. Access Control & Encrypted Communications.
Access Control. To mitigate unauthorized access to co-clients data and IP in spatial multi-tenant setting, a DMA engine can be implemented in the FPGA shell to allow valid memory accesses only and to isolate the data of the different virtual FPGAs [6], [7], since IOMMU matches memory accesses per physical FPGA only. Whereas in case the FPGA is accessed through the network, the network interface controller implemented in the FPGA shell is augmented with VLAN-tagging to isolate the packets of different virtual FPGAs [9]. These solutions work as the first line of defense. Hence, for further protection of data at rest and in transit, encryption is used.

Elnaggar et al. [98] proposed to add an external hardware component to the FPGA, rather than the FPGA shell, secure authentication module (SAM) that manages the distribution of secret keys to clients and enforces access control to the FPGA-accelerated tasks based on these keys. A client then integrates the secret key in the SW-application and the FPGA bitstream and uploads them to the cloud. At runtime, SAM verifies that the SW-application and the requested FPGA-accelerated task belong to the same legitimate client by comparing the implanted secret key. If authenticated, the SAM sends both components a secret session key to encrypt their communications. Hence, the established trust relies on the security of the secret key given to the client and implanted in the bitstream. Therefore, a client bitstream must be protected, otherwise, the security of the entire scheme is compromised.

Encrypted Communications. Yazdanshenas et al. [99], [100] attempt to address the confidentiality of off-chip communication using encryption in the FPGA shell, the statically configured logic, for all incoming/outgoing traffic of a physical FPGA through its interfaces (PCIe, Network and off-chip memory storage). Another way to secure tenant communications is to add encryption core wrappers around the reconfigurable regions to encrypt the data before leaving the virtual FPGA. This prevent I/O bottleneck compared to the first option, where all tenants' data gets encrypted or decrypted in the shell. However, the choice between the two modes should consider the trust assumptions: opting for shell encryption implies trust in the shell and the CSP, whereas using an encryption wrapper per virtual FPGA protects the virtual FPGA traffic in an untrusted environment. The main issue with the latter mode is that clients must securely bring their own keys into the encryption wrappers, which we only see feasible if bitstream protection on cloud FPGAs is supported. Nevertheless, for both encryption modes, defenses against remote physical attacks should be in-place. The authors compared the cost of soft vs. built-in AES cores and recommended the integration of built-in cryptographic cores by the FPGA vendors. As an example, the authors proposed to leverage a hardwired Network-on-Chip (NoC) where the NoC can be used to connect these regions via encryption-enhanced routers [99], [100]. However, this would result in extensive architectural modifications to FPGAs and restrict the CSP choices, i.e., number and size of virtual FPGAs.

In summary, in temporal or spatial multi-tenant settings, data isolation or encryption of different tenants is performed either i) by the client logic implying client bitstream protection, or ii) by the FPGA shell implying trust in the shell and the CSP. This trust can be simply assumed as a perquisite or gained, e.g. by remote attestation. More discussions on this follow in § 9.1.

9. Discussion

In light of the analysis of the security concerns that would arise from the different cloud FPGA computing deployment models, we present next our insights on the open challenges and opportunities in establishing trust in FPGA-based computing in the cloud in § 9.1 and lessons learned from trusted computing on CPUs in § 9.2.

9.1. Trusted Computing on Cloud FPGAs

FPGA devices can be leveraged to establish the trust anchor required in different secure computation flows and services on the cloud, owing to their features and advantages to both software as well as hardwired static hardware. One prominent example is offering to run clients' workloads in a trusted execution environment that gets established in cloud FPGAs. This is analogous to running workloads in a trusted execution environment on a CPU in the cloud, which is being offered more by CSPs recently to provide confidentiality and integrity guarantees both in bare-metal instances [55], [56] and in virtual machines [54]. We envision the FPGA shell, or hypervisor, to set up the FPGA trusted execution environment or trust anchor, i.e., to configure the client IP bitstream through the internal configuration port on the FPGA. Moreover, the FPGA shell must ensure that the bitstream is actually configuring the intended/allocated reconfigurable region (virtual FPGA) and not overwriting the configuration of another virtual FPGA, if spatial multi-tenancy is enabled.

One significant open challenge here is establishing a trust anchor when the FPGA shell, also a bitstream itself, is untrusted. Therefore, a supporting infrastructure (architecturally in FPGA devices as well as within overlying protocols and toolchains) is required to establish trust in the FPGA shell in the first place. Then, it can provide clients with sufficient guarantees that their IPs and data remain contained in secure and untampered isolation during execution and at rest.

FPGA-TCB. We refer to the components on the FPGA that are required to establish a trust anchor as the FPGA Trusted Computing Base (FPGA-TCB). This TCB would be required to verify the integrity of the FPGA shell and configure it on the FPGA, similar to secure boot on CPUs. The FPGA-TCB should also provide remote attestation of the FPGA shell, or the part of it that establishes and manages the FPGA trusted execution environment. Remote attestation would be required to provide assurance to the clients regarding the integrity of the deployed cryptographic primitives used to decrypt and verify their IP bitstreams and the protected access to plain-text bitstreams. To enable FPGA shell remote attestation and client bitstream decryption, the FPGA-TCB would also be required to handle secret session keys generation, exchange and management. Note that some FPGA vendors offer some of these requirements in their recent FPGAs, e.g., Intel Black Key Provisioning enables remote and secure provisioning of the AES root key in Stratix 10 [101].

Since the immutable FPGA-TCB cannot be compromised by malicious parties, it forms a root of trust in the FPGA. By leveraging remote attestation, the trust is extended to the FPGA shell, specifically to FPGA shell components that deal with client IPs and data, e.g., decryption and configuration management logic as well as the logic that prevents plain-text bitstream readback through the different configuration ports.[5] In [102] a first step towards the establishment of trust anchor on cloud FPGA is made.

This calls for a comprehensive analysis of the *minimal* FPGA-TCB components (both hardware and software) to provide these required security guarantees. This includes investigating efficient secure key and certificate provi-

5. Readback property enables reading configuration memory content (LUTs and programmable connections) to support the implementation of error correction for reliability.

sioning as well as the minimal hardware primitives (e.g., cryptographic processors or cores and their verified side-channel-resilient implementations). Furthermore, which of these primitives must be hard-wired into the FPGA, and which can be provided as configurable logic? How much hardware-software co-design is needed and what are the ideal design choices therein? How can debug interfaces, such as the Joint Test Action Group (JTAG) interface among others, be provided securely to still enable cloud clients to debug their FPGA development without threatening the confidentiality of other IPs and computations running on the FPGAs? Last but not least, how can we provide bitstream validation while still preserving client IP protection where the bitstream is encrypted? Various usability/performance/security trade-offs are involved within these open questions, all of which require a comprehensive and systematic treatment and analysis.

Remote Physical Attacks & Defenses. On another front, remote physical attacks demonstrate and emphasize why *secure* spatial multi-tenant FPGA computing is challenging. Nevertheless, in quest for maximum utilization and as FPGA devices' capacities keep scaling, spatial multi-tenancy is becoming inevitable. Thus, it becomes imperative to develop mechanisms that provide the required future-proof security assurances that are unique to these emerging FPGA computing scenarios. Runtime defenses, presented in § 8.2, require the implementation of protection logic in the FPGA shell or in a client's allocated reconfigurable region to protect the victim logic against a specific leakage source, e.g., power leakage. Such defenses do not only consume non-negligible resources but also require the clients to have FPGA hardware design expertise in order to implement them and find the best trade-off between security and resources overhead. Whereas other runtime defenses require the implementation of configurable sensors and noise generators, which are considered malicious and would be prevented by a bitstream validation process.

Fundamentally, the key enabler of these attacks is that clients can freely configure their (malicious) logic on the FPGA fabric. Developing mechanisms for effectively scanning and validating the clients' logic for malicious circuit signatures seems to be the intuitive approach to prevent malicious logic from being configured on the FPGA in the first place. While recently proposed stand-alone tools for detection of malicious circuits and suspicious behavior show promising results, they detect known malicious circuits only, but provide no guarantees for future zero-day attacks. Moreover, these tools, deployed by the CSP, work at the netlist level and require reverse engineering of clients' bitstreams. This is especially a challenge when the bitstreams are encrypted for purposes of IP protection, as pointed out above.

Therefore, FPGA toolchains may need to additionally cater for providing such security guarantees in cloud deployment settings, e.g., by extending them with particular design rule checks to enforce certain constraints on routing (e.g., to eliminate potential crosstalk across co-located tenants) and checking for potentially malicious design primitives before generating the bitstream.

9.2. Trusted Computing on CPUs: Takeaways

Being the more established and classical computing setting in the cloud, various lessons can be drawn from CPU-based computing. These can be taken into account in proactively assessing the security challenges and requirements of FPGA-based computing in the cloud, as well as adapting defenses that can also apply for FPGAs and drawing analogies where possible. In the following, we discuss the most common examples of security threats in CPU-based computing and draw the analogy to FPGAs.

Software Runtime Attacks. Analogously to TEE infrastructures and SDKs in CPU-based computing, trust assumptions in FPGA SDKs are to be examined and questioned. The critical security implications of a memory corruption vulnerability, if found in the SGX SDK, have been demonstrated in [103], where the authors show how a vulnerability can be exploited to craft a return-oriented-programming attack chain in the Trusted Runtime System (tRTS) code handling enclave entry. Similarly, such vulnerabilities can be exploited to load unintended accelerator tasks on the FPGA or even inject malicious accelerator tasks [98]. Therefore, SDKs and toolchains for FPGA development, whether at the client's end or hosted by the CSP in the cloud, need to be scrutinized and tested for such vulnerabilities and hardened accordingly, such that trust assumptions regarding them are sufficiently justified.

Microarchitectural Attacks. The recent outbreak of microarchitectural attacks in CPU-based computing also motivates an analogous investigation in FPGA-based computing. Identical counterparts of attacks such as Spectre and Meltdown [104]–[109] are not feasible on FPGA devices since they exploit side effects of specific microarchitectural features, e.g., speculative execution, that simply do not exist on FPGA fabric. However, attacks that generally exploit shared resources, e.g. caches and other memory buffers (store, load, etc.), would principally be also feasible on FPGA devices in one form or another, if they were shared without isolation among FPGA co-tenants. We already discuss in more detail in § 5 how different types of cache attacks would also be feasible on FPGA devices, and which defenses would similarly apply. In principle, besides caches, all shared microarchitectural/hardware/logic resources by co-tenants on an FPGA device constitute a potential security threat that require a thorough security information flow analysis under the assumed threat model of the deployment settings in question. They may leak information on the execution of one tenant to another via an exploitable side channel, unless appropriate defenses, e.g., flushing, partitioning or randomization, are applied.

Power Drop & Power Leakage Attacks. CPU vendors provide software-accessible interfaces to dynamic voltage and frequency scaling to control the supply voltage and the operating frequency of different CPU components (cores, last-level caches, etc.) at runtime and are accessible only to privileged software. Recent attacks exploit these interfaces to maliciously drop the supply voltage [110], [111] or increase operating frequency [112] to precisely induce faults in the operations of victim applications to reveal confidential data, e.g., secret keys. Some CPU vendors further provide software-accessible interface to power measurements of individual components (cores, uncore, DRAM, etc.). This interface has been exploited in [53] to infer executed instructions and their operands in a victim application to extract secret data. Similarly, such attacks have already been shown feasible on FPGAs, although by unprivileged malicious co-tenants in § 7 and § 6. Nevertheless, in an untrusted environment, such attacks can also be launched by the privileged FPGA shell that comprises power viruses or sensors. Defenses mitigating these attacks on CPUs usually involve patches to disable the interfaces, which is not directly applicable for FPGA devices, since the attacks stem from configuring malicious logic on the FPGA fabric. To mitigate these attacks on FPGA devices, both the FPGA shell bitstream as well as the clients' bitstreams should be validated and vetted for malicious logic before configuration.

In summary, valuable lessons can be drawn from CPU-based computing for FPGA-based computing in the cloud; a rigorous and systematic security analysis of the underlying fabric and TCB early on is crucial for the derived security guarantees for all that lies on top of that. Furthermore, for FPGAs, unlike CPUs, the *configurable* fabric is a double-edged sword. On one hand, it allows seamless updating and patching of hardware and vulnerabilities therein. However, it also poses an even larger attack surface if exposed to malicious parties to configure with complete freedom. Therefore, providing techniques to vet and police configuration bitstreams while preserving their confidentiality is the key open and non-trivial challenge for FPGA-based trusted computing.

10. Conclusion

In this paper, we systematically visited the different deployment models of FPGA-based cloud computing, and highlighted their adversary models and required security guarantees. The threat landscape is diverse and spans different levels of the deployment stack. We categorized and analyzed the different feasible attacks in FPGA-based cloud computing with a focus on remote physical attacks. We further classified state-of-the-art countermeasures against remote physical attacks into preventive and runtime defenses and discussed their deployment and security guarantees. Bitstream validation, a preventive mechanism, is more comprehensive and effective against the different threats, whereas current runtime defenses are each tailored to detect or mitigate individual threats.

We concluded with our insight into the open challenges in establishing trust in FPGA-based computing in the cloud and identifying the required minimal trusted computing base on FPGA devices required to provide clients with the relevant security and IP assurance guarantees. We also discuss some takeaways and lessons learned from CPU-based trusted computing and the security challenges therein, and draw potential analogies for FPGA-based computing. We emphasize that developing mechanisms to vet and validate protected (e.g. encrypted) configuration bitstreams is the key open and non-trivial challenge for FPGA-based trusted computing.

Acknowledgment. We thank the anonymous reviewers for their valuable and constructive feedback. This project was funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 CROSSING/236615297 and the Intel Collaborative Research Institute for Collaborative Autonomous & Resilient Systems (ICRI-CARS).

References

[1] M. Research, "Project Catapult," https://www.microsoft.com/en-us/research/project/project-catapult.

[2] Microsoft Research, "Project Brainwave," https://www.microsoft.com/en-us/research/project/project-brainwave.

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

[4] Y. Xu, O. Muller, P.-H. Horrein, and F. Pétrot, "Hcm: an abstraction layer for seamless programming of DPR FPGA," in *IEEE International Conference on Field Programmable Logic and Applications (FPL).* IEEE, 2012.

[5] S. Byma, J. G. Steffan, H. Bannazadeh, A. L. Garcia, and P. Chow, "FPGAs in the cloud: booting virtualized hardware accelerators with openstack," in *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM).* IEEE, 2014.

[6] F. Chen, Y. Shan, Y. Zhang, Y. Wang, H. Franke, X. Chang, and K. Wang, "Enabling FPGAs in the cloud," in *ACM Conference on Computing Frontiers.* ACM, 2014.

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

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

[9] O. Knodel, P. Lehmann, and R. G. Spallek, "RC3E: reconfigurable accelerators in data centres and their provision by adapted service models," in *IEEE International Conference on Cloud Computing (CLOUD).* IEEE, 2016.

[10] A. Khawaja, J. Landgraf, R. Prakash, M. Wei, E. Schkufza, and C. J. Rossbach, "Sharing, protection, and compatibility for reconfigurable fabric with amorphos," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI).* USENIX Association, 2018.

[11] D. R. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on FPGAs using valid bitstreams," in *IEEE International Conference on Field Programmable Logic and Applications (FPL).* IEEE, 2017.

[12] K. Matas, T. La, N. Grunchevski, K. Pham, and D. Koch, "Invited tutorial: FPGA hardware security for datacenters and beyond," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA).* ACM, 2020.

[13] 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 (TCHES)*, 2018.

[14] D. Mahmoud and M. Stojilović, "Timing violation induced faults in multi-tenant FPGAs," in *Design, Automation & Test in Europe Conference & Exhibition (DATE).* IEEE, 2019.

[15] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: remote power analysis attacks on FPGAs," in *Design, Automation & Test in Europe Conference & Exhibition (DATE).* IEEE, 2018.

[16] J. Gravellier, J.-M. Dutertre, Y. Teglia, and P. Loubet-Moundi, "High-speed ring oscillator based sensors for remote side-channel attacks on fpgas," in *International Conference on ReConFigurable Computing and FPGAs (ReConFig).* IEEE, 2019.

[17] O. Glamočanin, L. Coulon, F. Regazzoni, and M. Stojilović, "Are cloud FPGAs really vulnerable to power analysis attacks?" in *Design, Automation & Test in Europe Conference & Exhibition (DATE).* IEEE, 2020.

[18] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *IEEE Symposium on Security and Privacy (S&P).* IEEE, 2018.

[19] J. Gravellier, J.-M. Dutertre, Y. Teglia, P. L. Moundi, and F. Olivier, "Remote side-channel attacks on heterogeneous SoC," in *Smart Card Research and Advanced Applications*, S. Belaïd and T. Güneysu, Eds. Springer International Publishing, 2020.

[20] 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 (TCHES)*, 2020.

[21] I. Giechaskiel, K. Rasmussen, and J. Szefer, "Reading between the dies: cross-SLR covert channels on multi-tenant cloud FPGAs," in *IEEE International Conference on Computer Design (ICCD)*. IEEE, 2019.

[22] D. R. Gnad, C. D. K. Nguyen, S. H. Gillani, and M. B. Tahoori, "Voltage-based covert channels in multi-tenant FPGAs," *IACR Cryptology ePrint Archive*, vol. 2019, p. 1394, 2019.

[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 *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2018.

[24] G. Provelengios, C. Ramesh, S. B. Patil, K. Eguro, R. Tessier, and D. Holcomb, "Characterization of long wire data leakage in deep submicron FPGAs," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2019.

[25] I. Giechaskiel, K. B. Rasmussen, and K. Eguro, "Leaky wires: information leakage and covert communication between FPGA long wires," in *Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 2018.

[26] I. Giechaskiel, K. Eguro, and K. B. Rasmussen, "Leakier wires: exploiting FPGA long wires for covert-and side-channel attacks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2019.

[27] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "Measuring long wire leakage with ring oscillators in cloud FPGAs," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2019.

[28] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "Remote inter-chip power analysis side-channel attacks at board-level," in *International Conference on Computer-Aided Design (ICCAD)*. IEEE/ACM, 2018.

[29] I. Giechaskiel, K. Rasmussen, and J. Szefer, "C3APSULe: cross-FPGA covert-channel attacks through power supply unit leakage," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2020.

[30] S. Tian and J. Szefer, "Temporal thermal covert channels in cloud FPGAs," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2019.

[31] Intel, "Disaggregated servers drive data center efficiency and innovation," https://www.intel.com/content/www/us/en/it-management/intel-it-best-practices/disaggregated-server-architecture-drives-data-center-efficiency-paper.html.

[32] S. Biookaghazadeh, M. Zhao, and F. Ren, "Are FPGAs suitable for edge computing?" in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*. USENIX Association, 2018.

[33] K. Eguro and R. Venkatesan, "FPGAs for trusted cloud computing," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2012.

[34] B. Hong, H.-Y. Kim, M. Kim, T. Suh, L. Xu, and W. Shi, "Fasten: an FPGA-based secure system for big data processing," *IEEE Design & Test*, 2017.

[35] M. E. Elrabaa, M. Al-Asli, and M. Abu-Amara, "Secure computing enclaves using FPGAs," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2019.

[36] I. Kuon, R. Tessier, and J. Rose, *FPGA Architecture: Survey and Challenges*. now, 2008.

[37] D. Koch, *Partial reconfiguration on FPGAs: architectures, tools and applications*. Springer Science & Business Media, 2012, vol. 153.

[38] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *IEEE International Symposium on Computer Architecture (ISCA)*. IEEE, 2014.

[39] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, D. Firestone, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur *et al.*, "Configurable clouds," *IEEE Micro*, 2017.

[40] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H. Angepat, V. Bhanu, A. Caulfield, E. Chung *et al.*, "Azure accelerated networking: SmartNICs in the public cloud," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2018.

[41] H. Cloud, "FPGA accelerated cloud server," https://www.huaweicloud.com/en-us/product/facs.html.

[42] A. Cloud, "Fpga-based compute-optimized instance families," https://www.alibabacloud.com/help/doc-detail/108504.htm, 2019.

[43] Baidu, "Baidu cloud compute," https://cloud.baidu.com/product/fpga.html.

[44] Telekom, "Telekom ECS," https://open-telekom-cloud.com/en/products-services/elastic-cloud-server.

[45] C. Plessl and M. Platzner, "Virtualization of hardware-introduction and survey." in *ERSA*, 2004.

[46] A. Vaishnav, K. D. Pham, and D. Koch, "A survey on FPGA virtualization," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018.

[47] D. L. How and S. Atsatt, "Sectors: divide conquer and softwarization in the design and validation of the Stratix® 10 FPGA," in *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2016.

[48] Z. István, G. Alonso, and A. Singla, "Providing multi-tenant services with FPGAs: case study on a key-value store," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018.

[49] R. Kenny, "Soc FPGA hardware security requirements and roadmap," *Intel SoC FPGA Development Forum (ISDF16)*, 2016.

[50] P. Koeberl, "Multi-tenant FPGA security: challenges and opportunities," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2020.

[51] T. Trippel, K. G. Shin, K. B. Bush, and M. Hicks, "An extensible framework for quantifying the coverage of defenses against untrusted foundries," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2020.

[52] 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," *ACM SIGARCH Computer Architecture News*, 2014.

[53] M. Lipp, A. Kogler, D. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss, "PLATYPUS: software-based power side-channel attacks on x86," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021.

[54] Microsoft, "DCsv2-series VM now generally available from azure confidential computing," https://azure.microsoft.com/en-us/blog/dcsv2series-vm-now-generally-available-from-azure-confidential-computing.

[55] IBM, "Data-in-use protection on IBM cloud using Intel SGX," https://www.ibm.com/cloud/blog/data-use-protection-ibm-cloud-using-intel-sgx.

[56] Alibaba, "ECS bare metal instance," https://www.alibabacloud.com/product/ebm.

[57] 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)*, 2019.

[58] T. La, K. Mätas, N. Grunchevski, K. Pham, and D. Koch, "FPGADefender: malicious self-oscillator scanning for Xilinx UltraScale+ FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2020.

[59] S. Trimberger and S. McNeil, "Security of FPGAs in data centers," in *International Verification and Security Workshop (IVSW)*. IEEE, 2017.

[60] Y. Luo and X. Xu, "Hill: a hardware isolation framework against information leakage on multi-tenant FPGA long-wires," in *IEEE International Conference on Field-Programmable Technology (FPT)*. IEEE, 2019.

[61] Z. Seifoori, S. S. Mirzargar, and M. Stojilović, "Closing leaks: routing against crosstalk side-channel attacks," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2020.

[62] 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 *International Conference On Computer Aided Design (ICCAD)*. IEEE/ACM, 2019.

[63] M. Nassar, S. Bhasin, J.-L. Danger, G. Duc, and S. Guilley, "BCDL: a high speed balanced dpl for FPGA with global precharge and no early evaluation," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2010.

[64] A. Wild, A. Moradi, and T. Güneysu, "Glifred: glitch-free duplication towards power-equalized circuits on FPGAs," *IEEE Transactions on Computers*, 2017.

[65] Z. Chen, S. Haider, and P. Schaumont, "Side-channel leakage in masked circuits caused by higher-order circuit effects," in *International Conference on Information Security and Assurance (ISA)*. Springer, 2009.

[66] K. H. Boey, Y. Lu, M. O'Neill, and R. Woods, "Random clock against differential power analysis," in *IEEE Asia Pacific Conference on Circuits and Systems*. IEEE, 2010.

[67] T. Güneysu and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011.

[68] Y. Lu, M. P. O'Neill, and J. V. McCanny, "FPGA implementation and analysis of random delay insertion countermeasure against dpa," in *IEEE International Conference on Field-Programmable Technology (FPT)*. IEEE, 2008.

[69] A. Moradi and O. Mischke, "On the simplicity of converting leakages from multivariate to univariate," in *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2013.

[70] D. Jayasinghe, A. Ignjatovic, and S. Parameswaran, "Rftc: runtime frequency tuning countermeasure using FPGA dynamic reconfiguration to mitigate power analysis attacks," in *Design Automation Conference (DAC)*. ACM, 2019.

[71] G. Provelengios, D. Holcomb, and R. Tessier, "Characterizing power distribution attacks in multi-user FPGA environments," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2019.

[72] *Security Monitor Product Brief*, Xilinx Inc., 2019.

[73] *Intel FPGA Temperature Sensor IP Core User Guide*, Intel Corporation, 05 2018.

[74] 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 (TCHES)*, 2020.

[75] T. Kim, M. Peinado, and G. Mainar-Ruiz, "STEALTHMEM: System-level Protection Against Cache-based Side Channel Attacks in the Cloud," in *USENIX Security Symposium*. USENIX Association, 2012.

[76] Z. Zhou, M. K. Reiter, and Y. Zhang, "A Software Approach to Defeating Side Channels in Last-Level Caches," in *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2016.

[77] 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 *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016.

[78] Z. Wang and R. B. Lee, "New Cache Designs for Thwarting Software Cache-based Side Channel Attacks," in *IEEE International Symposium on Computer Architecture (ISCA)*. ACM, 2007.

[79] Y. Wang, A. Ferraiuolo, D. Zhang, A. C. Myers, and G. E. Suh, "SecDCP: Secure Dynamic Cache Partitioning for Efficient Timing Channel Protection," in *Design Automation Conference (DAC)*. ACM, 2016.

[80] D. Gruss, J. Lettner, F. Schuster, O. Ohrimenko, I. Haller, and M. Costa, "Strong and Efficient Cache Side-channel Protection Using Hardware Transactional Memory," in *USENIX Security Symposium*. USENIX Association, 2017.

[81] V. Kiriansky, I. Lebedev, S. Amarasinghe, S. Devadas, and J. Emer, "DAWG: A Defense Against Cache Timing Attacks in Speculative Execution Processors," in *IEEE Micro*, 2018.

[82] G. Dessouky, T. Frassetto, and A.-R. Sadeghi, "HybCache: Hybrid Side-Channel-Resilient Caches for Trusted Execution Environments," in *USENIX Security Symposium*. USENIX Association, 2020.

[83] D. Trilla, C. Hernandez, J. Abella, and F. J. Cazorla, "Cache Side-channel Attacks and Time-predictability in High-performance Critical Real-time Systems," in *Design Automation Conference (DAC)*. ACM, 2018.

[84] M. K. Qureshi, "Ceaser: Mitigating Conflict-based Cache Attacks via Encrypted-Address and Remapping," in *IEEE Micro*, 2018.

[85] M. Werner, T. Unterluggauer, L. Giner, M. Schwarz, D. Gruss, and S. Mangard, "ScatterCache: Thwarting Cache Attacks via Cache Set Randomization," in *USENIX Security Symposium*, 2019.

[86] Q. K. Zhu, *Power distribution network design for VLSI*. John Wiley & Sons, 2004.

[87] *7 series FPGAs PCB design guide*, Xilinx Inc., 5 2019, rev. 1.14.

[88] T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. Levin, T. Nguyen, and C. Irvine, "Moats and drawbridges: an isolation primitive for reconfigurable hardware based systems," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2007.

[89] *UltraScale architecture and product data sheet: overview*, Xilinx Inc., 8 2019, rev. 3.10.

[90] G. Piret and J.-J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and KHAZAD," in *International workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2003.

[91] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, 2011.

[92] S. Tian, W. Xiong, I. Giechaskiel, K. Rasmussen, and J. Szefer, "Fingerprinting cloud FPGA infrastructures," in *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. ACM, 2020.

[93] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in FPGAs," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2013.

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

[95] A. Duncan, A. Nahiyan, F. Rahman, G. Skipper, M. Swany, A. Lukefahr, F. Farahmandi, and M. Tehranipoor, "SeRFI: secure remote FPGA initialization in an untrusted environment," in *VLSI Test Symposium (VTS)*. IEEE, 2020.

[96] M. Hoffmann and C. Paar, "Stealthy opaque predicates in hardware-obfuscating constant expressions at negligible overhead," *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, 2018.

[97] J. Kataria, R. Housley, J. Pantoga, and A. Cui, "Defeating cisco trust anchor: A case-study of recent advancements in direct FPGA bitstream manipulation," in *USENIX Workshop on Offensive Technologies (WOOT)*. USENIX Association, 2019.

[98] R. Elnaggar, R. Karri, and K. Chakrabarty, "Multi-tenant FPGA-based reconfigurable systems: attacks and defenses," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019.

[99] S. Yazdanshenas and V. Betz, "Improving confidentiality in virtualized FPGAs," in *IEEE International Conference on Field-Programmable Technology (FPT)*, 2018.

[100] S. Yazdanshenas and V. Betz, "The costs of confidentiality in virtualized FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[101] *Intel Stratix 10 Device Security User Guide*, Intel Corporation, 10 2020.

[102] S. Zeitouni, J. Vliegen, T. Frassetto, D. Koch, A.-R. Sadeghi, and N. Mentens, "Trusted configuration in cloud FPGAs," in *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), to be published.* IEEE, 2021.

[103] A. Biondo, M. Conti, L. Davi, T. Frassetto, and A.-R. Sadeghi, "The guard's dilemma: efficient code-reuse attacks against intel SGX," in *USENIX Security Symposium.* USENIX Association, 2018.

[104] P. Kocher, J. Horn, A. Fogh, D. Genkin *et al.*, "Spectre attacks: Exploiting speculative execution," in *IEEE Symposium on Security and Privacy (S&P).* IEEE, 2019.

[105] C. Canella, D. Genkin, L. Giner, D. Gruss *et al.*, "Fallout: Leaking data on meltdown-resistant cpus," in *ACM Conference on Computer and Communications Security (CCS).* ACM, 2019.

[106] M. Schwarz, M. Lipp, D. Moghimi, J. Van Bulck, J. Stecklina, T. Prescher, and D. Gruss, "Zombieload: Cross-privilege-boundary data sampling," in *ACM Conference on Computer and Communications Security (CCS).* ACM, 2019.

[107] S. van Schaik, A. Milburn, S. Österlund, P. Frigo, G. Maisuradze, K. Razavi, H. Bos, and C. Giuffrida, "Ridl: Rogue in-flight data load," in *IEEE Symposium on Security and Privacy (S&P).* IEEE, 2019.

[108] S. van Schaik, A. Kwong, D. Genkin, and Y. Yarom, "SGAxe: how SGX fails in practice," https://sgaxeattack.com, 2020.

[109] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "Sgxpectre: Stealing intel secrets from SGX enclaves via speculative execution," in *IEEE Symposium on Security and Privacy (S&P).* IEEE, 2019.

[110] Z. Kenjar, T. Frassetto, D. Gens, M. Franz, and A.-R. Sadeghi, "V0LTpwn: attacking x86 processor integrity from software," in *USENIX Security Symposium.* USENIX Association, 2020.

[111] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: software-based fault injection attacks against intel SGX," in *IEEE Symposium on Security and Privacy (S&P).* IEEE, 2020.

[112] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: exposing the perils of security-oblivious energy management," in *USENIX Security Symposium.* USENIX Association, 2017.

[113] J. S. Wong, P. Sedcole, and P. Y. Cheung, "Self-measurement of combinatorial circuit delays in FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2009.

[114] K. M. Zick and J. P. Hayes, "Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2012.

[115] D. R. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "Analysis of transient voltage fluctuations in FPGAs," in *IEEE International Conference on Field-Programmable Technology (FPT).* IEEE, 2016.

Appendix A. Hardware Circuits for Remote Physical Attacks

A.1. Reconfigurable Sensors

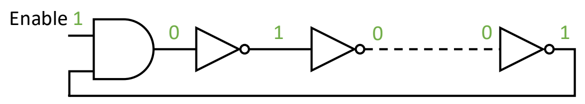
While on-die sensors to monitor FPGA supply voltage and temperature are typically available on recent generation of FPGAs, their response time and sampling frequency are typically not sufficient to capture short time-constant effects such as voltage transients. Therefore, fine-grain sensors that can measure the impact of physical properties (e.g., voltage and temperature variations in power distribution network, aging, etc.) of the different regions on the reconfigurable logic have been proposed [113], [114]. Two prominent examples of configurable sensors, which are implemented in the FPGA fabric that can perform self-measurement of their path delays are the ring oscillator and delay-line based sensors. Since propagation delay in circuits is highly affected by process, voltage and temperature (PVT) variations, measurement of combinatorial [6] propagation delays provides a means of measuring fine-grained changes in operating conditions and process variations. Nevertheless, calibration or post-processing steps might be required to suppress the undesired effects [114], [115].

Ring Oscillator (RO) Sensor consists of an AND gate (enabler) and an odd number of serially-chained inverters, referred to as stages, where the output of the last stage is fed back to the input of the AND gate (along with an enable signal), thus forming a *combinatorial loop* circuit. Further, the AND gate and one inverter may be combined in NAND gate. Other *sequential* RO sensors are the latch-based RO and the flip-flop based RO [94]. When a RO circuit is enabled, the RO output signal alternates between '1' and '0' continuously. To measure RO *oscillation frequency*, the RO output is used to trigger a counter circuit. The counter value is sampled at a user-defined rate, such that changes in the counter value between successive samples of the same RO reflect changes in operating conditions. RO sensors used in the different remote physical attacks differ in the number of stages, the counter design (binary or non-binary) and whether sequential elements (latches or flip-flops) are used. A classical ring oscillator (RO) is shown in Fig. 4a. All inverters but one can also be replaced with digital buffers as shown in Fig. 4b. Both RO designs Fig. 4a and Fig. 4b are *pure combinatorial* logic circuits, whereas the latch-based RO and the flip-flop (FF) based RO, shown in Fig. 4c and Fig. 4d respectively, are sequential. RO/oscillation frequency is inversely proportional to the delay of a signal traversing the feedback loop through the AND gate and chained inverters/buffers. Therefore, the oscillation frequency is dependent on the number of inverters/buffers and their propagation delays.

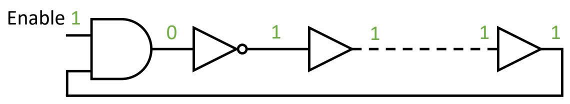
Delay-Line Sensor consists of a chain of n buffers, through which a clock signal propagates. Propagation delay is measured by how far the clock propagates through the buffers within a fixed time frame using a time-to-digital convertor (TDC) circuit. Latches are tapping into the connected buffers such that the propagation delay in the delay-line is measured by observing the boundary between all '1's latches and all '0's latches. The n latches are sampled at some user-defined rate and the obtained value is encoded into a $log_2(n)$-bit binary value, such that changes in the binary value between successive samples reflect changes in operating conditions. To minimize its area overhead (latches and encoder complexity), delay-line buffers can be split into initial and observable, such that only the observable buffers are connected to the latches.

Zhao and Suh [18] examined the trade-off between RO and delay-line based sensors in terms of sampling

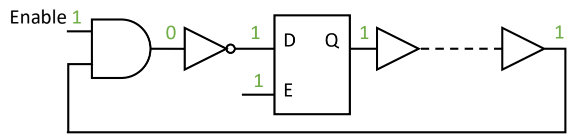
6. Combinatorial circuits consist only of wires and logic gates i.e. no memory elements such as flip-flops.



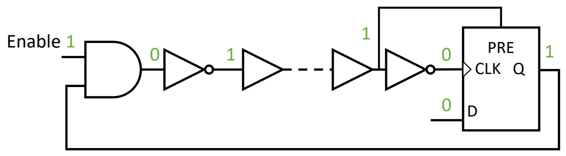
(a) Pure combinatorial RO design of odd number of inverters/stages.



(b) Pure combinatorial RO design of one inverter & even number of buffers.



(c) Sequential RO design with a D-latch [27].



(d) Sequential RO design with a D-FF [27].

Figure 4. Different RO Designs.

frequency, accuracy, resolution and complexity (area overhead and ease of implementation). In general, determining which on-chip sensor to deploy and its sampling frequency highly depends on the physical properties to be measured.

A.2. Reconfigurable Power Viruses

Examples of configurable circuits that have been used for power dissipating in the literature are: RO-based, FF-based and PIP-based power viruses. By toggling a large group of such power viruses for enough time, voltage drops (both transient and steady-state drops) can be induced in the power distribution network of the FPGA. **FF-based** power virus consists of a D flip-flop (D-FF), whose output is connected to the input of an inverter, while the inverter output is fed back to the input of the D-FF [115]. **PIP-based** power virus is constructed by leveraging programmable interconnect points (PIPs) in Xilinx FPGAs. These are programmable transistors used to connect inputs and outputs of I/O blocks and CLBs into the routing network in the FPGA fabric. Unused PIPs and wires in the FPGA fabric resemble a high capacitance, such that frequently charging and discharging them result in transient voltage drop and overshoot in the PDN [93].

An on-chip power dissipating circuit can be precisely controlled to induce power glitches within a specific time-frame. Moreover, such power viruses can be activated for long enough time to heat the FPGA. Therefore, power dissipating circuits can be leveraged to mount power fault attacks, as well as on-chip noise generators for mitigating side-channel attacks, as we describe later in more detail.

Appendix B. Details on Attacks Exploiting Crosstalk

Attack [23]. The key of the final round of the AES-128 can be recovered byte by byte. Then the encryption key can be obtained by inverting the key schedule. This requires the attacker and victim circuits to have 16 transmitter-receiver pairs of neighboring long wires, with one pair for each byte of the final round key in case of a high-throughput AES implementation with a datapath of 128-bit. Consequently, this is reduced to one pair of neighboring long wires if the AES has an 8-bit datapath.

The crosstalk effect increases for longer transmission pair and lower operating frequencies of the victim logic, which directly influences the number of encryptions required to recover the key. A minimum of 217 encryptions are required to recover the key for an AES core running at 10KHz, whereas at a higher operating frequency of 4MHz a 1.5 million encryptions are required. Interestingly, this crosstalk effect can also be leveraged to build an adjacency map for long wires within each channel in the FPGA [24], which is considered a proprietary information.

Attacks [25] and [26]. The crosstalk effect is demonstrated in the context of integrating encrypted or obfuscated third-party FPGA IP designs from multiple vendors in the same design [25]. The propagation delay of the receiver's long wire is highly influenced by how long a neighboring victim long wire carries a signal '1' within a sampling period. Thus, the counter value, i.e., the frequency of the RO connected to the receiver, indicates the hamming weight of the transmitted bits over the victim long wire within that sampling period. A sliding-window scheme is proposed to sample the counter at overlapping periods to extract the value of each transmitted bit of a secret key. By subtracting two overlapped hamming weights, information on the first and last bits of the corresponding windows are revealed. Assuming that the victim long wire is transmitting the whole key bits sequentially, and using a sliding-window of size w bits and a secret key of size N bits, where N is a multiple of w, i.e., $N = nw$, the authors compute the probability of recovering the key:

$$P = \left(1 - \frac{1}{2^{n-1}}\right)^{w} \tag{1}$$

This implies that the probability of key recovering is higher for larger keys (asymmetric vs. symmetric keys) and smaller window sizes. Equation 1, however, does not take into account noisy hamming weight measurements due to operating conditions (transmission frequency) and layout of transmission pair (length of transmission lines and distance between them). The sliding-window approach is applied to 32-bit keys in [26]. The authors demonstrate that using a sliding window of size $w = 4$ bits, 98.4% of the 32-bit key can be recovered assuming a single bit stays constant on the transmitter for 1.28 μs, i.e., 780 KHz signal frequency, and a transmission pair of at least 1-segment long wire. Moreover, it requires connecting four counters (equal to the window size) to the snooping circuit, where these counters are sampled at overlapping periods of 5.12 μs (4 $\times$ 1.28 μs).