Journal of Hardware and Systems Security (2024) 8:94–112
https://doi.org/10.1007/s41635-024-00148-4

Enabling Secure and Efficient Sharing of Accelerators in Expeditionary Systems

Arsalan Ali Malik[1] · Emre Karabulut[1] · Amro Awad[1] · Aydin Aysu[1]

Received: 25 July 2023 / Accepted: 17 March 2024 / Published online: 8 May 2024

Abstract
The addition of FPGAs in the cloud is an emerging effort to support acceleration and performance with the flexibility of logic reprogramming. The underlying logic per unit area of the FPGA chip has multiplied, making it challenging for a single-user design to utilize completely and efficiently. Major service providers (such as Amazon, Alibaba, and Baidu) are moving toward a shared FPGA model that allows system designers to share the chip fabric either *spatially* or *temporally*. This virtual partitioning of FPGAs is comparable to the expeditionary systems that also adhere to the same principle of *sharing chip fabric* among multiple tenants. These tenants have the potential to execute any untrusted application on this shared hardware, which is a serious cause for concern in expeditionary systems. For instance, a tenant can deploy malicious circuits that compromise the *confidentiality*, *integrity*, and *availability* of its fellow tenants. In this paper, we investigate the threat landscape and propose mitigation strategies for multitenant FPGAs. We assess threats to the *confidentiality* of users' critical data that are novel to the FPGA-as-a-Service (FaaS) framework. We present a defense mechanism for cloud FPGAs that verifies the *integrity* of tenants. In order to safeguard multi-tenant FPGAs from denial-of-service (DoS) attacks, our secondary defense mechanism promptly identifies malicious tenants and notifies the cloud orchestrator, thereby ensuring *availability*. We offer a comprehensive, all-in-one solution designed to defend and mitigate various threats faced by users in multi-tenant cloud FPGAs (in the public domain). The same principles apply to expeditionary systems with SWAP-constrained devices where multiple (potentially untrusted) applications share the same hardware. The proposed solution is thus adaptable and extendable to both public cloud service providers and expeditionary systems with private cloud infrastructure. The results show that the proposed work offers (i) safe-and-secure isolation of tenants, (ii) run-time access policy updates, and (iii) resilience against DoS attacks.

Keywords Remote-attack · FPGA · Security and safety · Multi-tenant cloud FPGAs · Safe reconfiguration

1 Introduction

In contrast to Application-Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) are a distinct integrated circuit type that provides *logic* reprogramming choices even after fabrication. FPGA's flexibility in design reconfigurability, higher throughput, and moderate power consumption offer the best of both hardware and software. Modern FPGAs nowadays come equipped with runtime reconfiguration capabilities that allow for the (virtual) division of FPGAs into *dynamic* and *static* regions [1]. The dynamic region of the FPGA can be reprogrammed using runtime reconfiguration while the static portion of the device keeps functioning normally. This allows users to load/unload designs onto remote FPGAs, benefiting from a plethora of services, such as hardware-backed acceleration [2], genomics research [3]. Increased computational power and runtime reconfigurability have allowed FPGAs to penetrate the cloud computing domain successfully.

The FPGAs can be quite powerful in terms of the resource and computation power they provide. For instance, Amazon

✉ Arsalan Ali Malik
aamalik3@ncsu.edu

Emre Karabulut
ekarabu@ncsu.edu

Amro Awad
ajawad@ncsu.edu

Aydin Aysu
aaysu@ncsu.edu

[1] North Carolina State University, Raleigh, NC, USA

EC2 F1 allows clustering up to eight Virtex UltraScale+ VU9P FPGAs together, each providing 1, 182 Look-Up-Tables (LUTs), 75.9MB Block RAM (BRAM) blocks, and 6, 840 DSP engines. A single-user's design is unlikely to occupy each of these resources at all times fully. Thus, an FPGA of this magnitude may share these resources (*multi-tenancy*) with various tenants to justify the associated costs, such as power, hardware resources, and energy. *Multi-tenancy* is the sharing of resources by a number of independent tenants who coexist and operate in a shared space.

FPGA-as-a-Service (FaaS) has emerged as a cloud model that lowers the service cost of an FPGA through two means: ① It eliminates the user's need to purchase and set up physical FPGAs. ② It presents the vendors with a new revenue stream that is highly remunerative. Sharing/partitioning of the FPGAs in the cloud can be employed based on two models, namely **spatial** and **temporal** tenancy. In spatial tenancy, the tenants occupy *separate* physical parts of the FPGA fabric at the *same* time. By contrast, temporal tenancy allows the sharing of the *same* physical parts of FPGA in *different* time intervals.

FPGAs nowadays are also being employed to create technologies that can aid expeditionary forces and mission-critical systems [4, 5]. FPGA's usage in such state-of-the-art expeditionary systems is for a variety of reasons, including (i) ease of reconfiguration, (ii) reduced time to market, and (iii) fault tolerance. As a logical next step, multi-tenant FPGAs may be used to expand the capabilities of such systems. For instance, mission-critical systems accessing a centralized cloud can use FPGAs to offer system security and expedited cryptographic services such as encryption, hashing, and secure video feed sharing. Expeditionary systems must defend assets, all the while fighting against adverse scenarios. This is why such systems are often targeted by *attackers* seeking ways to compromise their—Confidentiality, Integrity, and Availability. Consequently, securing all such expeditionary systems from present and future (potential) threats is critical.

In an exhaustive literature study, we have identified three key obstacles that multi-tenant FPGAs face, threatening the tenant's *confidentiality, integrity,* and *availability* (CIA). (i) In a shared environment, *malicious* tenants can launch attacks that steal the user's *confidential* data [6, 7]. Therefore, CSPs are in need of countermeasures that resist such attacks [8]. (ii) Vendor-provided support for safe and secure communication among the tenants is either flawed [9, 10], or non-existent [11]; lacking the ability to identify and allot FPGA space to a tenant whose identity and *integrity* can be validated. (iii) PR is a power-hungry and time-consuming process that needs security and power efficiency optimizations. An *attacker* can exploit the power consumption aspects of PR to cause system failure and threaten *availability*. A significant effort exists in academia solely to overcome the costs associated with PR to increase its efficiency in terms of power and execution time [12–20]. However, these works overlook the security aspects of PR operation for multi-tenant FPGAs.

This paper focuses on the obstacles described above and provides concrete implementations to that end. The paper also outlines a future roadmap and discusses strategies to address the involved challenges in Section 6. Thus, this paper apprises cloud-tenant providers and consumers about reconfiguration challenges and mitigation strategies that ensure the safety and security of tenants in a multi-tenant environment. Moreover, we examine the FaaS deployment paradigm for expeditionary systems and associated applications, as well as the challenges tenants may face while creating trustworthy computing in the cloud. We next outline three main contributions:

- *Confidentiality:* In multi-tenant FPGAs, tenants sharing the same chip fabric are prone to remote side-channel attacks. Although side-channel attacks (SCA) on neural networks are a well-known effort [7, 21, 22]; we demonstrate an attack that can break *confidentiality* for the *first* time in the context of FaaS[1]. The results show that the proposed methodology provides a confidence rate of 99.9% without physically probing the FPGA.
- *Integrity:* To verify the tenant's data *integrity* and prevent access violations in a shared channel, a cloud orchestrator must ascertain that all tenants adhere to a well-defined set of rules. We expose that existing commercial tools fail to enforce this, and we propose a defense mechanism that ensures the tenant's *integrity* in a multi-tenant environment by preventing unauthorized modification of tenants' data by employing a *strong* access policy. The proposed defense monitors each tenant via a *four-stage* mechanism to circumvent challenges pertaining to functionality, scheduling, safety, and security.
- *Availability:* Multi-tenant FPGAs tend to share the power distribution networks (PDN) among their tenants. As a result, high activity in one tenant may disable or affect the timeliness of another tenant's computation. For the *first* time, in this study, we develop power profiles by analyzing the power usage of PR that a *malicious* tenant might exploit to launch a DoS attack. We also present a novel low overhead, tuneable power monitoring defense that can detect and defend against attacks that threaten resource *availability* in multi-tenant FPGAs.

[1] Previous research has focused on either obtaining model hyperparameters or extracting secret cryptographic keys, remotely. In contrast, our work focuses on extracting model parameters remotely from an ML accelerator.

- *Defense Strategies:* To fend against the attacks on *confidentiality*, *integrity*, and *availability*, we propose a mechanism that combines the three aforementioned mitigation strategies, having a minimal resource footprint. We built this mechanism based on our prior work to facilitate CIA in FaaS model [7, 9, 23]. The results show that our strategies have low resource overhead and are easy to adopt in multi-tenant FPGAs due to the use of the standard AXI interface.

Organization The paper is organized as follows. Section 2 gives the necessary prerequisite knowledge to grasp the proposed work. This section concludes by establishing a threat model for the subsequent sections. Section 3 presents the existing research in this area and highlights its limitations. Section 4 provides the novel contributions of this work and outlines the steps in detail to adopt security and power awareness for a multi-tenant environment. Section 5 evaluates the proposed work in real-work applications and provides its efficacy through results. Finally, Section 6 motivates the readers regarding the need for further work in this domain, concluding the paper in Section 7.

2 Background

This section provides the information necessary to grasp the inner workings of PR [1], which is a key enabler for multi-tenancy, followed by the threat model used in evaluating this study.

2.1 PR Interfaces of Xilinx Zynq

The Xilinx Zynq SoC was recently added to the 7-Series family of FPGAs. The Zynq devices are unique in the aspects that it consist of two programmable sections on a single chip: (i) programmable logic (PL), which is a traditional FPGA, and (ii) processing system (PS), which extends the support of ARM microprocessor to an FPGA. The PL can host multiple partitioned designs (a.k.a *tenants* in the cloud environment). These tenants can be dynamically loaded onto the chip using programming interfaces such as JTAG, MCAP, ICAP [24], and, most recently, PCAP [25].

The PCAP interface is limited only to Zynq SoCs and is controlled through PS. ICAP and PCAP interfaces are physically multiplexed; thus, at any given time, only one of them can access the PL fabric. Xilinx also provides a safety mechanism that prevents switching of control between ICAP and PCAP. Therefore, the CSP must use one of these interfaces to support multi-tenancy. ICAP requires manual instantiation by the user logic, whereas PCAP, which is a part of the PS, has no such limitation. The choice of ICAP and PCAP impacts the device's power consumption. This study only focuses on designs incorporating the ICAP interface for the reasons described at the end of Section 5.1.3.

2.2 AMBA-4 AXI Interconnect

Tenants (a.k.a cloud clients) may often need to communicate with each other by establishing a new or utilizing an existing channel. The provision of such an interface was once a complex problem. To overcome this issue, Xilinx standardized an interface that must be adapted to connect different intellectual properties (IPs). The AMBA-4 AXI is a communication bus interface designed by ARM [26] and adopted by Xilinx to connect IPs with the help of a simplified interconnect. AXI interconnect helps reduce the congestion of communication signals while *isolating* the IP's critical path from one another. In fact, Xilinx, by default, wraps its commercial IPs with the AXI interconnect interface, and for Amazon F1 cloud FPGAs, designers are mandated to connect their IPs to cloud DDR memory units using this interface. Therefore, in a multi-tenant FPGA, the AXI interconnect should also suffice as a reliable and robust means of communication, ensuring both performance and scalability.

The AXI interface works on the principle of a master-slave. The communication is point-to-point in nature. Generally, a design requires only one master to communicate with one or more slaves. For designs involving multiple masters, an additional AXI crossbar module that provides arbitration and connectivity among IPs must be included. Xilinx provides this arbitration using address mapping methodology. A master's access cannot exceed the slave's address space assigned to it. If numerous master IPs map to a single slave IP, the *access control policy* defines the *isolation* and access control method among distinct masters.

2.3 Threat Model

Multi-tenant FPGAs are not exempt from the notions outlined by the information security principles, the CIA triad. This study examines the realm of FPGA multi-tenancy in light of three key concepts defined by the CIA triad.

First, we present a remote side-channel attack that can compromise the *confidentiality* of tenants. *Second*, we show the *lack* of tenant integrity mechanisms in vendor-provided solutions, followed by a novel approach that preserves the *integrity* of tenants in cloud FPGAs by preventing *unauthorized* modifications. *Third*, we dive into the PR operation, highlighting concerns that can impede the *availability* of FaaS by inducing a DoS attack. In addition, we also offer a defense mechanism to thwart such attacks.

We follow the threat model of the prior works to highlight the need for information security principles in cloud FPGAs [7, 9, 23]. For each threat model, the CSP is assumed

to be trustworthy and committed to granting tenants equitable and lawful access to the FPGA fabric allocated to them. The *attacker* is assumed to have complete control over configuring the design space allocated to them[2]. Following is a summary of these threat models:

- **Confidentiality**. The tenants follow the *spatial* tenancy model in which an *attacker* can be any tenant colocated on the chip fabric. The *attacker* targets ML applications to break *confidentiality* and extract high-value parameters such as *weights* [7]. The *attacker* does not have physical access to the FPGA. However, the *attacker* can remotely create a DUT on the same FPGA and use it to do computations. The *attacker* either knows or has the means to know the inputs fed to the model and the model's hyperparameters.
- **Integrity**. This threat model covers concerns related to both *spatial* and *temporal* tenancy use cases under the following assumptions: (i) Two spatial tenants that share an interconnect bus could try to modify each other's data to violate the access policy and undermine the channel's *integrity*. (ii) Tenants have access to both on and off-chip memory resources. (iii) In the case of *temporal* tenancy, the cloud provider may switch out two tenants at various periods. Consequently, the new tenant may use the same physical interface port as the prior tenant.
- **Availability**. The threat model tackles the prospective usage of multi-tenant FPGAs with *spatial* tenants. The *attacker* has access to the shared PDN and, thus, can perform experiments to characterize the PDN boundaries. The *attacker's* primary aim is to cause DoS and shut down the FPGA. The *attacker's* secondary goal involves breaking the determinism of PR execution delay [9]. In summary, the attacker's long-term goal is to deny *the availability* of resources to legitimate users.

The scope of this work currently does not cover concerns related to timing, EM, fault attacks, ML-based defense, or fair resource allocation among tenants. These are orthogonal efforts, left as a future effort, and discussed briefly in Section 6.

3 Literature Review

Recent qualitative studies reveal that physical attacks have grown diverse, targeting the *confidentiality* [7], *integrity* [27–29], and *availability* [9] of tenants. At the core, these attacks have been made possible because the victim and the attacker (i) share the same FPGA fabric (although logically isolated), (ii) are free to configure the space allocated to them with any (malicious) logic of their choice, and (iii) lack a mechanism that discourages and prevent unauthorized access of tenant's data. The advancement of technology and the ever-growing sophistication of attacks have plunged cloud computing environments into an era of new threats, *e.g.,* multi-tenant FPGAs, GPUs, CPUs, *etc.,* are shown to be equally vulnerable to side-channel attacks (SCA) and exploits as their offline counterparts.

SCA belongs to a class of attacks that aim to steal system secrets by employing physical attacks on the device. Classic cryptanalytic attacks target the algorithm's mathematical basis, whereas SCA targets the actual implementation. Therefore, in addition to the algorithm's flaws, the attacker seeks to exploit the system's vulnerabilities. For SCA, where physical access to the device was (generally) deemed necessary to capture the device's behavior, recent attacks have relaxed these restrictions [7, 30]. Thus, a check and balance mechanism for multi-tenant FPGAs is imperative. We now review these attacks, the proposed countermeasures, and their limitations in the context of information security principles.

Confidentiality of data is a critical and well-studied area in the information security domain. The demand for *data confidentiality* is utmost for multi-tenant FPGAs due to tenants' *shared* nature of physical *resources*. To this end, resource *isolation* has been investigated as one of the means to ensure data privacy. A recent study shows that a number of attacks fall under the umbrella of privilege violations, looking to bypass or breach *isolation* [31]. These attacks aimed to access components not in their address space, such as memory [27, 32]. These attacks were facilitated by the fact that defenses for on-chip memory (e.g., BRAM) sharing are lacking compared to their equivalent off-chip counterparts (e.g., DRAM), which has been an active area of research for multitudes [33, 34].

Xilinx provides a defense mechanism for expeditionary and mission-critical system's on-chip resource isolation called *Isolation design flow* (IDF) [35]. IDF provides logic segregation by incorporating *fences* into design logic. *Fences* are areas in FPGA fabric through which no *unauthorized* user or routing logic can pass. Recent studies have found IDF effectiveness to be inadequate against fault attacks [36, 37]. Researchers have also sought ways based on the IDF's ideology to create *active fences* capable of detecting and defending against voltage-based SCAs.

These studies identified three key problems with IDF. (i) IDF provides isolation using *fences* that can only be placed at design time. This is problematic for multi-tenant FPGAs as which tenant may occupy the FPGA cannot be

[2] Vendors/CSPs may try to prevent this capability by detecting/restricting certain types of *malicious* circuits. However, these defenses can be bypassed by adversaries, as shown in Section 4.3.

established at compile time. Moreover, tenants often swap in and out of the cloud environment. (ii) IDF limits logic placement around the fence. In a multi-tenant FPGA, this implies employing complex design rules and wasting precious PL resources on account of each tenant that enters/exits the cloud space. (iii) IDF and PR cause design rule check (DRC) conflicts when enabled side-by-side, rendering it unreliable for multi-tenant FPGAs that rely highly on PR. Although recent work has found a way around the last problem, their solution is limited to specific classes of FPGAs and does not provide design assurance [38].

The reduced size of integrated circuits generates a capacitive cross-talk channel on the FPGAs interconnect, which can be exploited to extract the AES secret key [39]. This direct breach of data's *confidentialilty* must be eliminated in multi-tenant FPGAs by maintaining adequate *isolation*. A countermeasure against such an attack is proposed in [40], which employs an obfuscation technique on the long wires in a design to lower side-channel leakages. It is important to note that leaks are not *completely* removed and continue to pose a genuine danger to tenant's data *confidentiality*.

The lack of vendor-endorsed mechanisms for data *confidentiality* has also compelled academia and industry to develop their custom solutions besides *isolation*, *e.g.,* implementation of encryption in the FPGA shell using the static logic region to secure off-chip communication [41]. The problem with such an approach is the secure communication overhead through FPGA interfaces such as PCI–e, Ethernet, etc.

An improvement on the previous technique [41] was recently proposed that secures tenant communications with minimal overhead and trust assumptions [42]. The proposed solution involves placing encryption core wrappers over reconfigurable areas instead of all interfaces, encrypting data before it leaves the virtual FPGA. Compared to the prior approach, where all tenants' data is encrypted or decrypted in the shell, the former prevents I/O bottlenecks. However, the trust assumptions should be considered when choosing between the two modes: shell encryption *assumes* faith in the shell and the CSP, whereas utilizing an encryption wrapper per virtual FPGA protects virtual FPGA communication in an untrusted environment. The main challenge with the second method is that clients must securely transport their secret key(s) into the encryption wrappers, which is only possible if cloud FPGAs allow bitstream encryption [43].

Measuring the power consumption of a victim by placing voltage sensors near the victim's logical boundary is a growing effort to break data *confidentiality*. These leaky traces can then be collected remotely and processed locally for a successful DPA attack. Recent work used high-speed voltage transient sensor that is coupled with a delay line to sense voltage fluctuations in a remote FPGA's PDN [44]. These fluctuations were recorded by a *malicious* tenant (attacker) while an AES-128 core was running on another tenant (victim).

Building upon this, an improved attack that requires fewer traces to conduct correlation power analysis (CPA) has also been proposed [45]. The attack is also effective for higher-end FPGAs such as the Xilinx Ultrascale+ FPGA [46]. The secret key associated with the AES-128 core was recovered in just 30 attempts, with a success rate of 42%. Public key cryptographic cores are also vulnerable to this class of attacks. The square-and-multiply step in RSA-1024 has also been targeted using an RO-based voltage sensor to perform simple power analysis (SPA) [30]. Using just 20 RO-based sensors, the collected traces were enough to reveal the RSA's private key. The designer of this exploit argues that the attack is equally successful for processes running on CPUs sharing the same PDN.

Integrity refers to the safety of data by preventing unauthorized modifications. In multi-tenant FPGAs, a tenant can eavesdrop and modify the data of their fellow tenant to cause damage or financial loss. Regardless of the *attacker's* intent, the tenant relies on the CSP to ensure the *integrity* of their data in a shared environment. The channel on which tenants may communicate with each other must be reliable, robust, and trustworthy.

Xilinx-provided AXI interface uses an address-mapping-based *weak* mechanism to facilitate the tenants'/IPs' communication. Several security and privacy issues are highlighted for this MPSoC FPGAs' bare-metal, unprotected memory access model, such as lack of *integrity* and trust assurances [47]. To circumvent these, Xilinx recommends using Memory Protection Units (XMPUs) and Peripheral Protection Units (XPPUs) [48, 49]. Unfortunately, these protection configurations cannot be extended to a multi-tenant setup due to the (i) absence of any *isolation* functionality within the PL itself, (ii) *ineffectiveness* against remote SCAs [7, 27, 30, 50], and (iii) lack of functionality that ensures the *integrity* of each tenant.

Additionally, Xilinx FPGAs are unable to offer memory isolation for on-chip FPGA memories, *e.g.,* BRAM. While previous works have provided memory virtualization to provide off-chip memory (DDR) isolation for multi-tenant FPGAs [51–53], these methods do not address the isolation problems with on-chip FPGA memories. Xilinx FPGAs support BRAM connections through a single AXI crossbar interconnect, which may be shared, thereafter, among several master IPs. As a result, via this BRAM connection, one tenant can access the data of another and make *unauthorized* modifications. A possible solution involves assigning a dedicated BRAM per tenant, but this approach results in *less* efficient resource use.

For the latest MPSoC FPGAs, Xilinx offers the support of ARM TrustZone, allowing users to group several IPs in a

design by marking them as either *trusted* or *untrusted*. This divides the IPs into two distinct groups, with the *trusted* group having access to both the *trusted* and *untrusted* groups. By contrast, the *untrusted* group has access only to the *untrusted* group. This management poses three problems. *First*, TrustZone's distinction can only be established at design time; hence, the prospective IPs must be known at design time. Predicting which tenant's IP will occupy the FPGA fabric in a multi-tenant context is extremely challenging. *Second*, the lack of dynamic updates in the access policy results in transferring the previous tenant's security profile onto the current tenant's profile (inheritance). *Third*, if a tenant's IP turns out to be *malicious*, there is no control mechanism to detect and blacklist such a tenant from reacquiring the same fabric space.

In summary, current multi-tenant FPGAs lack adequate mechanisms to verify *the integrity* and provide high-performance *isolation* for memory resources, processor cores, and IPs.

Availability of the compute-intensive resource is what the user is essentially paying for when renting cloud FPGAs. Security and privacy become less important to legitimate users if they cannot access the promised resource timely despite paying the high associated costs. This is why *attackers* often sought ways to disrupt the *availability* of resources, causing denial-of-service.

In a *spatial* multi-tenancy model, tenants share the power distribution network (PDN). This sharing of PDN has led to several attacks and exploits that measure and characterize the power consumption of the FPGAs. [9, 37, 54, 55]. These attacks aim to deprive the *availability* of resources in a shared environment by causing a power failure or creating a device malfunction.

To thwart such attempts, some vendors provide a tool that is only design-time configurable [56]. The tool functionality is limited to monitoring and reporting the system's power to the user, leaving the burden of taking appropriate action to the user. Multi-tenant FPGAs require a mechanism to detect and thwart such attacks requiring minimal tenant intervention and ensuring *availability*.

Attackers seek ways to force the system power's boundaries, derailing it into a "gray zone." These power fluctuations directly impact PR operation latency, breaking its determinism and denying *availability*. Such determinism is paramount in mission-critical expeditionary systems where system reliability may mean the difference between life and death [4, 5]. The prior work has accordingly modeled the importance of determinism and *availability* in PR [57, 58]. However, none of these earlier studies considered the voltage levels and their impact on their considered model. This work presents a specific power load that can break the determinism of the PR. Furthermore, we evaluate a recent threat model for the FaaS paradigm that shows how a *malicious* tenant can launch an *attack* that threatens *availability*.

In light of the considerations and limitations mentioned above, we now present our proposed work that mitigates the issues and shortcomings of the prior work while conforming to the information security principles.

4 Mitigation of CIA Issues for Multi-Tenant FPGAs

This section describes and details the attacks and defenses for the multi-tenant FPGAs to ensure *confidentiality*, *integrity*, and *availability* in the FaaS model.

4.1 Confidentiality of FaaS

Using physical side-channel attacks to steal machine learning (ML) models from embedded devices has become a new and growing concern. With the FPGAs growing richer in resources, it has become virtually impossible for a single user to use all of the FPGA resources. As a result, this may motivate a shift to cloud-based solutions offering FaaS. For a CSP providing multi-tenancy on the FPGA fabric may be more cost-effective, in which customers use different parts of the same FPGA fabric. However, several users sharing the same resources might result in unintentional and potentially serious security vulnerabilities like SCA.

Multi-tenant FPGAs were considered to thwart side-channel attacks naturally due to physical inaccessibility. However, recent works have shown the applicability of *remote* side-channel attacks on multi-tenant FPGAs that do not require physical access to extract secrets [7, 30]. Their work has demonstrated the presence of side channels between multiple tenants sharing the same chip fabric due to the shared PDN. Prior work has successfully exploited the shared PDN channel to launch a SPA attack to extract the secret key of AES [30]. We refer interested readers to the recent surveys that categorize literature on this topic [31, 59, 60].

This work shows a side-channel attack on neural networks (NN), specifically on binary neural networks (BNN). Although SCA on NN is a well-known effort [7, 21, 61, 62], we demonstrate this attack for the first time in the context of FaaS. We utilized time-to-digital (TDC) converters to extract the *confidential* and highly lucrative parameters; *weights*. We *assume* a tenant can have *malicious* intents while sharing the same FPGA fabric; the attacker knows the inputs to the deployed NN model and its hyperparameters. The sole focus of the attacker is on the unknown *weights* of the model without having physical access to the platform on which it is running. Extraction of these design *weights* is advantageous to

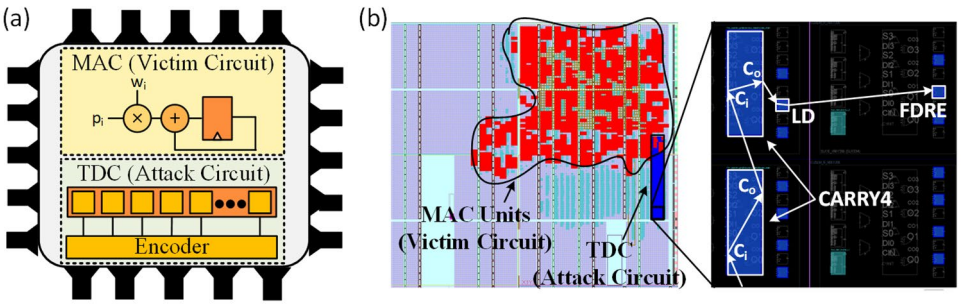

Fig. 1 The proposed remote attack scenario for our experiment **a** high-level view and **b** actual floor-plan of the TDC sensor and MAC unit is illustrated, respectively. The figure on the right shows a part of the TDC with the `CARRY4` primitives serially coupled through the carry-in/out ports to eventually feed an `LD` (latch) and `FDRE` (flip flop) cell

the attacker because of the excessive time and effort involved in the training and fine-tuning process.

For this attack to be feasible, we assume the case of *spatial* tenancy in which tenants share the chip and the PDN. The power variations are thus visible and measurable by all the tenants. This poses a serious threat as the power spikes generated due to confidential processing by tenant **X** can be exploited by tenant **Y**. Tenant **Y** can morph a power sensor using a TDC circuit whose frequency varies with the power level activity of tenant **X**. The TDC circuit can delay the propagation of a signal by varying the number of buffers to measure voltage. Alternatively, a malicious tenant **Y** can assemble a ring oscillator circuit with similar functionality. This will allow the *malicious* tenant to gain power traces of critical components while they perform some computation, *e.g.,* MAC unit in case of NN.

The multiply-accumulate (MAC) unit is known for its high power consumption in NN accelerators. Thus, the MAC is a point of interest while performing DPA attacks. We developed hardware that stores all inputs in an on-chip memory before processing them sequentially every cycle. Before the NN model starts execution, *weights* are transferred to on-chip memory. In BNN, the *weights* remain the same for an input value of 1, whereas, for an input value of 0, the 2's complement is computed. The results are then forwarded to the MAC that accumulates the intermediate computations to generate the final product for a single node per layer.

Figure 1a illustrates the conceptualized model that we created to conduct our experiments, whereas Fig. 1b shows the actual floor-plan view of our implemented circuit, consisting of the MAC units and the TDC sensors. Our remote power sensor is based on TDC consisting of a chain of buffers (a.k.a *delay chain*), storage elements to store the buffer output, and a priority encoder. A clock signal is fed to the input of TDC, which delays its propagation based on the number of active sequential buffers in its path. In parallel, the output of each buffer is *latched* operating on the same clock. Xilinx provides a carry chain hardware primitive with a low propagation delay called `CARRY4`. We utilized `CARRY4` to enhance the resolution of our TDC in its observable portion. `FD`[3] latch captures the number of buffers through which the rising edge of the clock traversed before changing its polarity and entering the inactive region. The outputs from the `FD` latch feed the TDC output register, which we implemented using the `FDRE` primitive. In the following step, we use a priority encoder to compress the output size from N to $\log_2 N$.

We also built supporting controller logic that ensures that the TDC circuit executes each time a MAC operation occurs in the tenant-of-interest. This provides an additional advantage of synchronizing the captured traces in time, which is challenging in physical and remote power attacks. The existing proposals have discussed these challenges in detail, which are orthogonal to this study. We refer interested readers to [44, 63] for a more thorough explanation. The accumulation performed by MAC after each operation results in a highly observable steep drop in the count value, which acts as a trigger point for the attack. The trigger registers the TDC output into a parameterized register file memory. The controller's depth and frequency govern the width of the capture window. Once the attacker has enough TDC counts, a DPA attack can be launched by exporting them to a remote PC to break the *confidentiality* of its fellow tenants.

Countermeasures The remote power attack works by monitoring the power activity of a remote tenant to extract key or secret data. We suggest defenses to fend against these attacks based on the prior works[4]. The attacks proposed

[3] We used Xilinx `FD` latch primitive to capture the output of buffer at each stage.

[4] Functional verification of these efforts is beyond the scope of this work. This work *does not* claim these defenses as our own, but lists them for reader convenience.

in the literature fall into two main categories: Hiding and Masking. To defend against such attacks, one must understand the *attacker's* intent and the targeted parameters.

4.1.1 Hiding Attacks and Defenses

In a multi-tenant setup, when a computation is performed, the effect is directly visible on the *unprotected* captured trace. This *unprotected* trace poses serious vulnerabilities that can be exploited to expose secrets. *Hiding* is a countermeasure proposed to defend against such vulnerabilities. *Hiding* reduces the reliance of the power consumption of the device on the intermediate variables processed within that device. In order to prevent information about the intermediate variables from being revealed in the power trace, *hiding* seeks to either randomize the power usage *or* to make it constant. This is generally achieved by: (i) reducing the signal-to-noise (SNR) ratio at electrical levels [37], (ii) amplifying the noise level by introducing additional noise sources in the design, *e.g.,* ROs, BRAM collisions, shift registers [64] (iii) clock randomization that spreads the side-channel information temporally [64], (iv) using dual-rail precharge logic [65], duplicating or inverting the core logic [66], to balance the power consumption.

Further examples of the defenses include: (i) adapting the software-based approach that creates different replicas of an IP using an automated approach to ensure tenant's *confidentiality* [67], (ii) a quantitative defense framework that uses static and dynamic frequency scaling to manage the clock frequency of FPGA while it hosts an application [68], (iii) an isolated wave dynamic differential logic (IWDDL) design flow that separates the direct and complementary circuit paths to resist DPA attacks in the hamming distance power model [69], (iv) domino logic array style based power aware implementation that predicts the power level of an operation based on the input vectors to generate compensating power that balances or lowers the peak power of the circuit [70], etc.

We refer interested readers to recent papers that capture the attacks and defenses on *hiding* in depth [8, 37, 44, 71–76].

4.1.2 Masking Attacks and Defenses

The second defense to protect multi-tenant systems against remote side-channel attacks is *masking* which divides internal values into various shares. Only when all shares are combined the secret value is revealed. *Masking* technique uses random numbers to make the intermediate variables independent of the intermediate variables in the algorithm. *Masking* techniques are shown to be susceptible to higher-order attacks, *e.g.,* higher statistical moments [77]. However, such attacks require significantly more traces to be successful.

An example of a *masking* method is AGEMA [78], which equips novice engineers and hardware designers with the tools to generate efficient masked designs from unprotected designs. AGEMA offers processing methods that convert an unsecured design into a secure one, thereby speeding up and protecting the concealment of hardware. BoMaNet is another effort that extends the *masking* technique to the neural networks [21]. Using secure hardware primitives and boolean masking, the authors addressed the challenges related to masking integer addition.

The designing phase of the masking techniques is prone to errors, and this is where the most complexity lies. To combat this, a hybrid approach that designs and verifies such masked circuits is proposed [79]. Their proposed solution provides heuristics and optimized algorithms along with quantitative verification of the program. We direct the readers to recent works that further discuss these defenses in detail [27, 30, 78, 80, 81].

4.2 Integrity of FaaS

A secure communication channel and tenant identity verification in multi-tenant FPGAs are essential in preserving the *integrity*, e.g., the need to verify the integrity of a tenant in cloud FPGAs arises when a tenant initiates an unauthorized modification or causes a link conflict by creating a deadlock on an interconnection bus. In the absence of tenant's *integrity* verification and safety mechanisms, there is nothing stopping a *malicious* tenant from manipulating data, eavesdropping, or causing neighboring tenants harm. Unfortunately, the vendor's and third-party solutions support in this domain is either inadequate [82] or (almost) non-existent [10].

Xilinx's implementation of AXI interconnect is proprietary and close-sourced, limiting direct modifications. The solution itself is only *design-time* configurable making it unsuitable for multi-tenant FPGAs running dynamic applications. Tenants in cloud environments require a solution that is *run-time* configurable, has low latency, and offers higher throughput. An ideal solution must be adaptive to both *spatial* and *temporal* cloud models. Moreover, the model should restrict tenants sharing the chip fabric from eavesdropping on their fellow tenant resources, such as memory and interconnect.

The inevitable failure of Xilinx's AXI Crossbar IP security policy is due to the assignment of a constant master ID to each slot that connects to a master IP (a tenant). The crossbar tags an AXI request issued by a tenant to organize AXI traffic. These *hard-coded* IDs are assigned at compile time; therefore, a cloud orchestrator cannot change them at runtime. This allows the reallocated tenants to gain access and cause unauthorized modifications to other tenants' data.

Cloud FPGAs users' need for a *run-time* reconfigurable solution is due to the fact tenants are swapping in and out of the cloud; therefore, the security configuration/policy that is only *design-time* configurable is of limited benefit. A dynamically updateable profile can ensure timely detection and defend against persistent malicious attackers. Moreover, a fine-grained dynamic control mechanism can help avoid the wastage of FPGA resources, ensuring efficient utilization, low system latency, and improved throughput.

Countermeasure Traditional cloud computers establish tenant's access permissions and perform verification at the *application layer* to guarantee tenant's data isolation [83]. This allows them to perform user authentication and request authorization before its execution. Such methods of isolation do not easily adapt to cloud FPGAs. Moreover, the authors in [84, 85] found weaknesses that can bypass such a security check. Their proposed attacks jeopardize data stored in the cloud since application-level security only offers a single level of defense. Therefore, we present a hardware-based solution that informs the CSPs and hardware engineers on how to establish a communication infrastructure for cloud FPGAs. Furthermore, this work provides significant improvements to the prior work [10], adding safety and security elements to cloud FPGAs.

To ensure tenant's data *integrity* in multi-tenant FPGAs, we propose Safe-and-Secure AXI (SS-AXI) that offers run-time management of the tenant's security configuration under both *spatial* and *temporal* settings. Unlike prior work [10] that limits the security configuration of a particular tenant against a specific policy, SS-AXI is not limited by such constraints. SS-AXI provides finer-grained access control on memory in an effort to reduce resource wastage. The proposed solution has a resolution of the double word (32-bits) for on-chip memory (BRAM) that allows for more efficient tuning of access policy.

Contrary to the requirement of previous work [10] to wrap a tenant in a trusted execution logic that incurs communication delays and reduces throughput, SS-AXI is not subject to any such restrictions.

The block diagram for our hardware design is shown in Fig. 2. Using Verilog RTL, we developed our own AXI crossbar interface that supports the AXI-Lite and AXI4-Full interfaces and has the following major components:

- **Master/Slave arbiters** are responsible for AXI traffic monitoring originating from master(s) and slaves(s), respectively.
- **Orchestrator** provides security configuration updates and runtime management capabilities. The granularity of the orchestrator is configurable at design time.

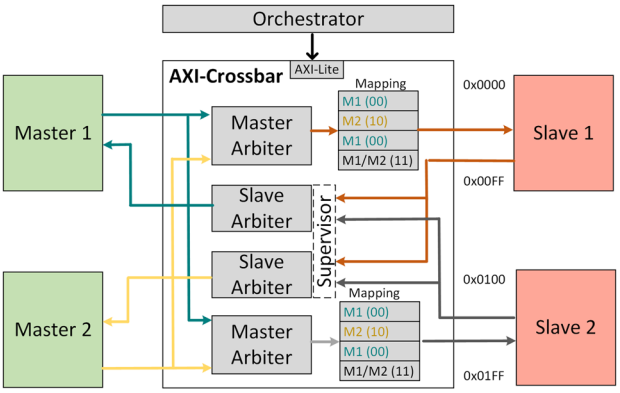

Fig. 2 The proposed Safe-and-Secure AXI (SS-AXI) crossbar design that manages traffic flow between the master and slave IPs. The orchestrator is capable of dynamic security authorization updates, and the mapping table supports safe resource sharing across master IPs

- **Mapping table** contains mappings for the bus master and slave resources and their relevant access policies.

The resource mapping mechanism in our design enables the secure mapping of a single slave to several tenants (master IPs). The mapping table granularity is customizable during design time. However, the SS-AXI orchestrator can also dynamically change the mapping table contents to update and modify tenants' access ranges once they are configured on the FPGA as part of the shell. This represents a significant advancement compared to prior work [10].

4.2.1 Master/Slave Arbiters

Figure 3 illustrates a simplified view of the proposed SS-AXI. A request generated over the AXI interface goes through all four stages depicted in the figure before reaching

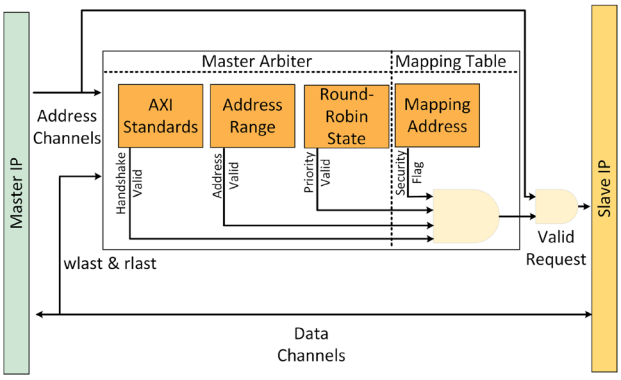

Fig. 3 There are four AXI check unit levels in the proposed design. This resolves functionality, scheduling, efficiency, safety, and security concerns in AXI traffic

its final destination. The *first* stage ensures that the generated request obeys the AXI interface standards. The *second* stage prevents functional error or access violations over memory address space, ensuring safety. In case of an access violation, the request is dropped, and the channel is kept busy. For requests originating from a master,[5] we also keep track of the *wlast* and *rlast* flags to ensure safe transactions over the AXI interface. The *third* stage addresses the challenges related to AXI scheduling conflicts. We have implemented a round-robin scheduling mechanism to facilitate back-to-back AXI requests.

The *fourth* stage's purpose is two-fold: (i) to provide efficient resource utilization and (ii) to preserve security. Vendor-provided crossbar IP [11] and prior work solution [10] support isolation by limiting one master to one slave. For a master that may need to connect to more than one slave, data isolation support is not available. The proposed work offers the flexibility to choose between an exclusive master or a shared communication interface (among multiple masters) without compromising isolation.

4.2.2 Orchestrator

The primary role of the orchestrator is to manage and update the security register configuration of our indigenous SS-AXI crossbar via its AXI-Lite port at runtime. In turn, these security registers manage mapping table and arbiter configurations. In the absence of transactions on the crossbar, the orchestrator can also configure the design configuration of the AXI crossbar. Using the AXI-Lite interface over the AXI-Full interface ensures simplicity and low resource overhead.[6] The latency of a configuration update varies between 96 to 3471 clock cycles, subject to the choice of resource-sharing granularity, software workload, and the use of the shared-vs-dedicate AXI smart connect bridge. We implemented our orchestrator on the PS side and assumed it to be *trusted*. A more sophisticated hardware-based orchestrator design that ensures the root of trust is also possible; however, that is an orthogonal effort.

4.2.3 Mapping Table

The mapping table allows sharing of a particular slave among one or more masters. It does so by maintaining an address table that checks the access range of each IP. The table ensures that each IP adheres to the orchestrator's defined access policy and isolation requirements. We also incorporate a `supervisor` module that watches over slave arbiters. `Supervisor` ensures that no new response is sent to a slave's second request until the master handles the first request. This is done to prevent dead-lock scenarios for requests that may go out of order to ensure safety.

[5] AXI burst read/write operations.

[6] AXI-Full interface is also supported, we prefer AXI-Lite for design simplicity.

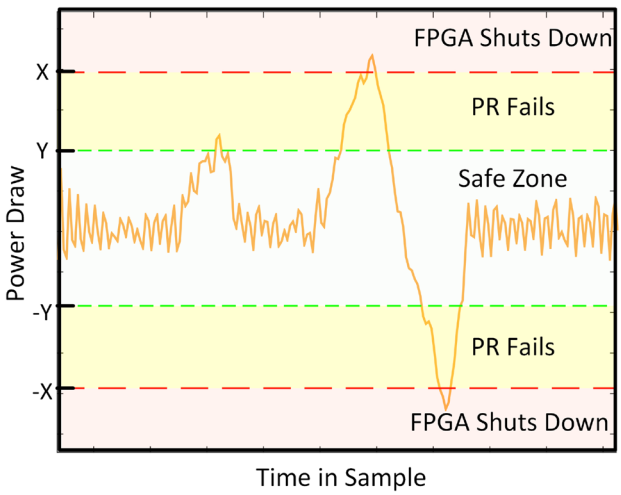

Fig. 4 Conceptualized power margins for safe PR in cloud FPGAs. The FPGA operates properly between the power ranges $[-X, X]$, but due to PR's power-demanding nature, the FPGA can only execute PR requests within the range $[-Y,Y]$

Using SS-AXI interconnect in multi-tenant FPGAs, one can ensure that tenants only communicate with one another over a trusted channel. The tenant's requests are monitored and forwarded through the orchestrator. If a tenant request exceeds its privileges, it will be detected immediately. The orchestrator can allow or drop requests as per the access policy defined by the CSP. If necessary, the CSP may additionally ban a *malicious* tenant based on its tag to prevent further attacks from the tenant. The proposed design thus helps ensure the *integrity* of each tenant and the communication channel for multi-tenant FPGAs.

4.3 Availability of FaaS

In addition to the safety and security concerns raised by multi-tenant FPGAs, timely *availability* of resources to legitimate tenants is also of utmost importance. A *malicious* tenant can hide behind the facade of legitimacy, waiting to initiate DoS for requests generated by legitimate tenants or may try to damage the PDN of multi-tenant FPGAs in order to cause long-term damage [86, 87]. In this section, we present a scenario in which a *malicious* tenant threatens the *availability* of resources to legitimate tenants, followed by a defense mechanism that can fend against such attempts.

A cloud FPGA's theoretical power draw margins are depicted in Fig. 4. We propose that the power consumption levels may be classified into three regions: (i) a safe zone in which the FPGA may operate normally; (ii) a PR-fail

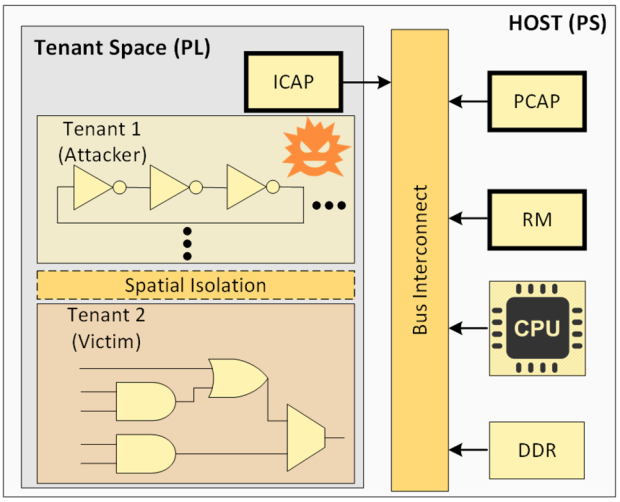

Fig. 5 A high-level Xilinx Zynq SoCs design model with multiple modules and peripherals attached over interconnect. An attacker can employ a power monitoring circuit e.g., ring oscillators (ROs), or TDC to push the limits of PR operation and induce complete PR failure

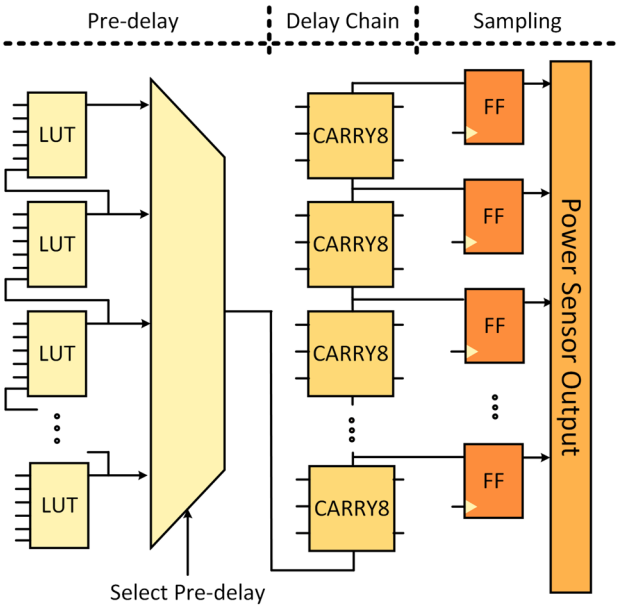

Fig. 6 Power sensor with a small footprint and self-tuning capability. The power sensor has three components: a self-tuneable pre-delay, `CARRY8` delay chain, and a sampler. Power sensors can observe the propagation delay of the chain to compute power consumption in real-time and support the AXI interface

region where a PR request leads to a device malfunction because of its higher power draw; and (iii) an unsafe region where the FPGA shuts down despite the power draw levels. Furthermore, we argue that there is a risk of increasing PR latency around the outer boundaries of the PR-fail zone, which is especially troublesome for real-time systems that need determinism.

Figure 5 presents a cloud model employing *spatial* tenancy. Tenant 1 represents the approach of an attacker trying to cause DoS for a victim, tenant 2. The tenant 1 design consists of a series of ROs that can generate a steep voltage drop upon activation. In the absence of a PR request, this malicious circuit is not a cause for an alarm, as the power consumption stays within a safe operating zone. When a PR operation is requested for a tenant 2, the power consumption of the multi-tenant FPGA will spike. These spikes are due to the PR being a power-hungry operation. This can result in a complete system failure causing the device to shut down and, in an extreme case, even burn out.

A smart adversary may take an even more cunning approach to evade detection. An attacker, tenant 1, can activate a minimum number of ROs, which keeps the power consumption in a safe zone, but marginally. A legitimate tenant's PR request (*e.g.,* tenant 2) will push power consumption beyond safe limits. In this scenario, the attacker may be driven by an objective to disable the FPGA and disrupt the determinism in the PR execution delay. Instead of just shutting off the FPGA right away, the attacker can cause unexpected power activity. This power activity generates variation in PR execution time, compromising PR's determinism. If a power monitoring system is in place, it can mistakenly identify and mark tenant 2 as a culprit, restricting it from gaining access to the cloud FPGA in the future. As a result, tenant 1 will be able to evade detection. Limiting the number of permitted tenants may also result in long-term financial loss for the cloud service provider.

Countermeasure We recently proposed a defense in [9] that monitors system power usage in run-time so that the CSP may pause/resume PR if needed. The proposed system provides this functionality while maintaining low overhead, high flexibility, and compatibility. The primary components of the defense offered are (i) a power sensor that can observe the tenant's power activity and (ii) a control logic that can calibrate and sample the power sensor's output. An alarm is raised to inform the control logic and cloud service orchestrator if the power sensor detects suspicious activity. The orchestrator can then pause the current PR operation until the system's power returns to a safe operating level or take any other defensive action in accordance with the CSP access policy.

Figure 6 presents the construction of the proposed power sensor. The power sensor uses FPGA primitive elements to sense the power variations. The control logic can tune the input delay by adjusting the pre-delay chain made of LUTs. These LUTs feed the long propagation delay chain made using the `CARRY8` chain. The output of the `CARRY8` chain is sampled using flip-flops (FF) at every clock cycle. The proposed solution is, thus, portable to any FPGA, supports

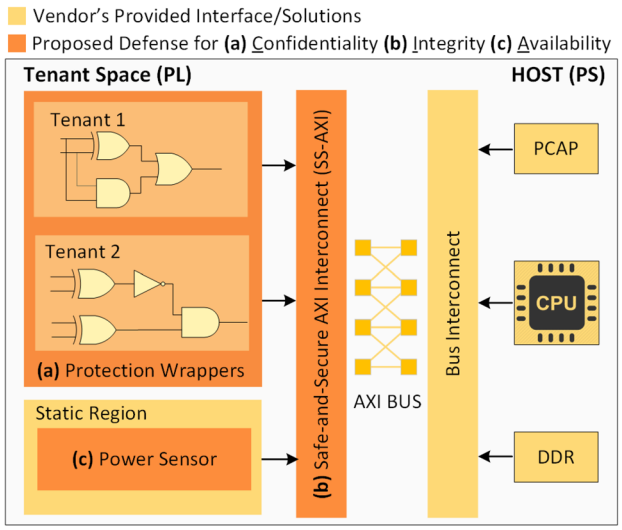

Fig. 7 The proposed defense mechanism conforms to the information security principles of the CIA. **a** Tenants are secured using logic-based protection wrapper (shown as an orange box) to protect their data's *confidentiality*. **b** The SS-AXI interconnect acts as a bridge between the AXI interconnect and tenants to verify *integrity*. **c** The power sensor monitors and reports the power consumption of each tenant to ensure *availability*

run-time calibration, and is fine-tuneable. The depth of the FFs is also parametric, allowing the cloud orchestrator to calibrate the resolution of the power sensor. The power sensor supports the AXI interface making its integration into complex design seamless.

4.4 Recipe for FPGA Multi-tenancy

The prior subsections provide the building blocks using which the proposed work builds an all-in-one framework for multi-tenant FPGAs. Section 4.1 promotes the need for data *confidentiality* in multi-tenant FPGAs by illustrating a practical remote side-channel attack on NNs. To ensure the *integrity* of each tenant occupying the FPGA spatially, we provided a Safe-and-Secure AXI (SS-AXI) interconnect that assures the tenant's identity for multi-tenant FPGAs in Section 4.2. The security policies of SS-AXI are run-time configurable, with low latency and higher throughput compared to prior work [10, 11]. Finally, in Section 4.3., we demonstrated an attack that can cause DoS and threaten resource *availability* in multi-tenant FPGAs.

An attacker can deploy a malicious circuit to push the limits of power consumption of cloud FPGAs. By exploiting the PR power consumption, an attacker can cause a DoS attack, depriving allocation of resources to legitimate tenants. We recommend employing our proposals to form a comprehensive framework that addresses expeditionary systems' security needs and safety challenges. The proposed work follows the guidelines of the CIA triad to provide a robust and scalable environment for mission-critical applications of expeditionary systems.

Figure 7 illustrates the usage of the proposed work for multi-tenant FPGAs. The proposed mechanism conforms to the three pillars of the information security principles. By using the countermeasures proposed in Section 4.1, tenants can choose among the variety of defenses to either *mask* their contents internally or be enclosed in a *logic-based* protection wrapper (shown as an orange box around tenants). These countermeasures prevent the abolition of the tenant's *confidentiality*. The SS-AXI interconnect uses the AXI interface to connect multiple tenants. The AXI traffic routed through SS-AXI helps verify the *integrity* of each tenant. The power sensor is placed in a static region of the PL that also uses the AXI interface to communicate. To assure *availability*, the power sensor can track and report on each tenant's power consumption.[7]

5 Results and Evaluation

This section evaluates our proposed design implementation and shows through results the effectiveness and resilience of the proposed design methodologies that ensure *confidentially, integrity*, and *availability* for multi-tenant FPGAs.

5.1 Evaluation Setup

We demonstrate our experiments on Xilinx Zynq FPGAs (ZCU104). The board features Zynq UltraScale+ MPSoC, which meets our requirement of PS and satisfies the required PL resources.

5.1.1 Breaking Confidentiality

We replicated the MAC layer of NN using Verilog RTL. The generated RTL was parsed through Xilinx Vivado v2021.1. A UART interface was included along with a MAC design to send and receive the inputs and outputs of MAC to the connected host PC, respectively. The PC uses a C# based application for communication and storing the captured TDC traces. In the MAC unit, the accumulation register stores the sum of each result as described in Section 4.1. We target these registers to launch a DPA attack to create a hamming distance-based power model. The rationale behind choosing this power model is that the power activity of an FPGA and the number of toggles in a register are related.

[7] TDC sensor can also be replicated multiple times for monitoring each tenant's power individually. This allows for more fine-grained control, but at the cost of more area.

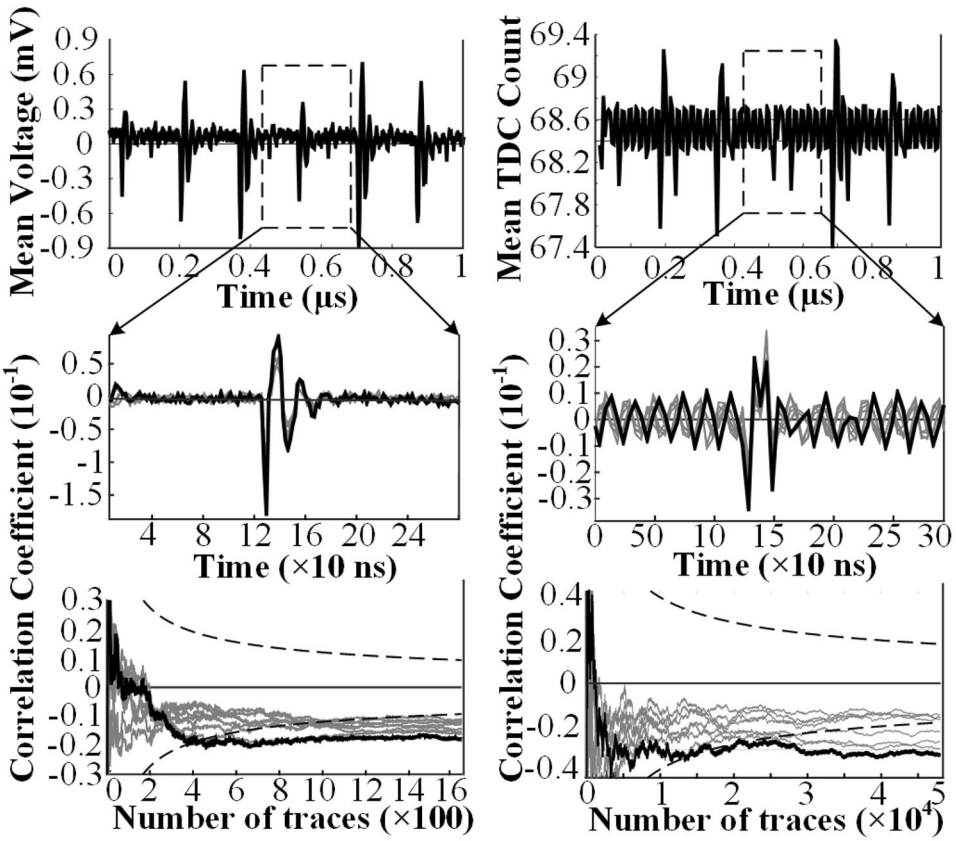

Fig. 8 The accurate (dark-colored) and inaccurate weight prediction (light-colored) in the performed DPA attack using the oscilloscope (figure's right-side) and remote power attack using TDC (figure's left-side) is shown. Beyond 400 traces, the Pearson correlation exhibits the 99.99% confidence interval (dotted lines) with actual power measurements and 25k traces with TDC measurements

The target implementation of this work is a BNN; hence, the *weights* can only vary between ±1. The attack is performed sequentially: extract the first n *weights* $\omega_0 - \omega_{n-1}$ by formulating a hypothesis on the n^{th} partial sum. The 2^n possibilities for $\omega_0 - \omega_{n-1}$ defines the size of hypothesis table as 2^n. Beginning with the n^{th} summation, we postulate the following n^{th} partial sum to extract the next n *weights*, and so on. The challenge in performing this attack is the low SNR ratio generated by a single MAC unit. This can be due to the small size of the MAC unit in comparison with complete encryption units e.g., AES, SERPENT, RSA, such as. Xilinx's use of a 28*nm* technology cell in the latest FPGA family may also be a factor in this low-power trace. To work around this challenge, we slightly modified the DUT by replicating the MAC unit 256 times to create a distinguishable power trace in the active vs. inactive (idle) state. The trace exhibits similar power peaks as a 10-round AES implementation. The replication was performed for simplicity to make the trace more visible, reducing the required number of traces. The attack is equally feasible regardless of this replication and does not affect the effectiveness of the attack itself. We refer interested readers to [7] to gain further insights regarding this design choice.

Figure 8 illustrates the results of the DPA attack on the first three *weights*. The left half of the figure shows the attack result using a TDC sensor, whereas the right half shows the attack on real measurements collected using an actual oscilloscope. The Pearson correlation shows the confidence of 99.99% after 400 and 25K traces for the oscilloscope and TDC acquired data, respectively. The lower traces are due to our multiple replications of the MAC unit.

5.1.2 Verifying Integrity

In Section 4.2, we presented a method to verify the *integrity* of tenants in multi-tenant FPGAs. We developed a custom AXI crossbar in Verilog RTL and tested its capabilities on real hardware. The testing was performed in a scenario where 'M' masters can communicate with 'N' slaves to emulate an M-to-N use case. The use of AXI interface makes the SS-AXI fully extensible and poses no bounds on the number of masters/managers and a number of slaves/sub-ordinates. Compared with Xilinx crossbar IP, we used 239 LUTs, and 61 registers, whereas Xilinx crossbar IP consumes 263 LUTs and 186 registers. Our solution reduces the resource usage by 1.1× and 3.1× for LUTs and registers, respectively. Additionally, the vendor IP has no built-in security mechanism, whereas our solution provides safety, security, and tenant *integrity* with no added latency.

The AXI interface of our designed crossbar IP was functionally tested using vendor-provided AXI verification IP [88]. We tested our designed solution for both AXI4-Lite

and AXI4-Full interface as a master and a slave using 16 tests written in System Verilog. We initially created two AXI-4 masters with separate settings to characterize various security violations that may occur in the cloud. The two masters also emulate a scenario of having/hosting two tenants in a multi-tenant FPGA. The orchestrator port of our crossbar IP was then linked to the UltraScale+ MPSoC, with two BRAM IPs acting as a slave. We used the PS section of the SoC to run software-based applications that emulated orchestrator tasks. We then proceeded to test those scenarios in which the master IPs were attempting to breach the security rules. The proposed solution allowed us to update the security configurations dynamically from the software-based program running on the PS.

Compared with a recent work [10], which provides runtime management capabilities, our solution is more efficient and ensures tenants' safety, security, and *integrity* in multi-tenant FPGAs without added latency. Moreover, the prior work adds an overhead in terms of wrapper logic that must be enforced for each tenant, which lacks fine-grained control over the interconnect configuration. By contrast, our solution allows a finer-grained control with *run-time* policy update capability.

5.1.3 Ensuring Availability

To ensure resource availability in a multi-tenant environment, we performed another experiment using the PS and PL sections of Zynq UltraScale+ MPSoC. We deployed our software application on PS, which had direct access to the AXI interconnect and PR access. On the PL side, we deployed an RTL-based *victim* tenant and an *attacker* tenant with *malicious* intent. The *malicious* tenant is a parametric chain of ROs that, when activated remotely, induces a sudden power drop. The circuit contains 52, 800 ROs in total that can be activated in steps of 100 ROs. No placement constraints were placed on the *malicious* circuit to make it truly location independent. To reduce the dependency on FPGA components (FFs, LUTs, etc.) for PR execution time, we built a resource area-dense PR IP that utilizes around 90% of the allowed region. Our system utilizes 100Mhz and 200Mhz clocks for the system operation and TDC-based power sensor, respectively. Activating the ROs simultaneously increases the chip temperature drastically; therefore, we performed our experiments in a series of steps to ensure uniformity. A total of 1000 PR operations were performed to observe the PR region's behavior and obtain a stable Gaussian distribution among test cases.

Figure 6 illustrates the three components that form our power sensor circuit. (i) a self-tuneable pre-delay having 32 LUTs, (ii) `128-CARRY8` length delay chain, and (iii) a sampler that uses 128 latches and registers. A control logic manages the power sensor and collects its output through

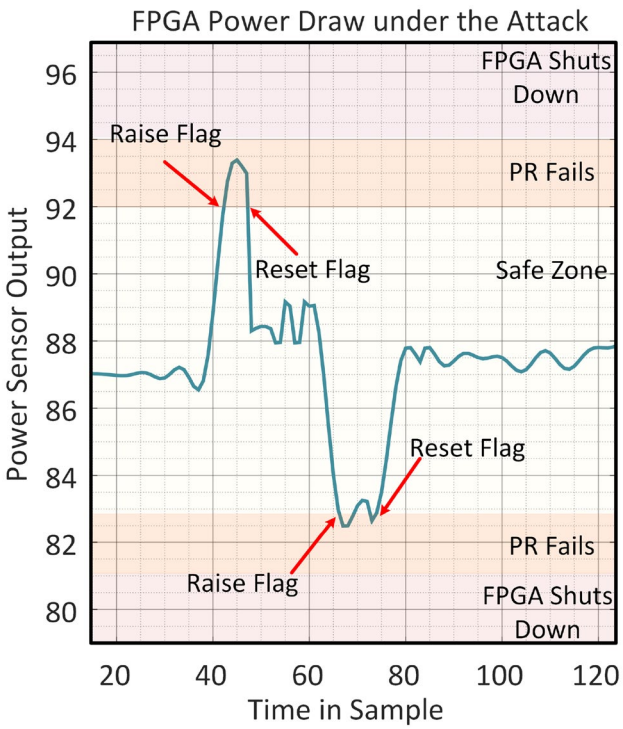

Fig. 9 The power sensor's output when FPGA is under attack by a malicious tenant. Postprocessing at the output of our digital power sensor is employed using a low-pass filter

the AXI-Lite interface. The control logic mainly consists of an FSM and registers with a resource utilization of 7276 LUTs and 4239 registers. The resource utilization reported is for one specific case. However, our control logic and power sensor are fully parametric, making it tunable and an ideal choice even for resource-constrained devices. We refer interested readers to [9] to understand the proposed TDC sensor architecture in-depth.

For ensuring the resource *avaiability* in multi-tenant FPGAs, our implementation first characterizes the power profile of the device. This gives us an upper and lower bound on the device's power sensitivity. The two attack scenarios we adopted are: create a sudden power drop by activating a large number of ROs, and gradually activating the ROs while creating a heavy power load on the PDN itself. We found the second attack methodology to be more effective in causing a device shutdown. To measure determinism failures, we intentionally drove the FPGA power consumption to the point where the FPGA can manage PR but is on the borderline of crashing (boundaries of the *Safe Zone* in Fig. 9). Then, we proceeded to measure the PR operation response time and its robustness.

We present our results in Fig. 9. The figure presents the output of our power sensor in the active and inactive state of 37, 400 ROs. Because rapid RO's activation only causes the voltage to undershoot, we gradually enabled 37, 400 ROs

before disabling them all at once to demonstrate voltage overshoot and undershoot in the same figure. We also mark three power margins in the figure to characterize our evaluation device as a result of our experiments. If the device's power is within the bounds of the *Safe Zone*, the device functions normally. If the power activity exceeds the bounds of the *Safe Zone*, we raise the alarm to inform the cloud orchestrator. The orchestrator then decides whether to process or pause the subsequent PR operations in the presence of alarm flags.

Setting the alarm and responding to the alarm has an overhead of 2 clock cycles, providing a small buffer zone in the device's power margins. If the cloud orchestrator decides to pause the PR operation, it should interrupt and terminate the transaction over the PCAP or ICAP interface. However, Zynq UltraScale+ MPSoC currently does not support interrupting the PCAP interface. In comparison, ICAP soft IP has interrupt support; therefore, for the proposed defense, we recommend utilizing ICAP over PCAP. It is also worth mentioning that in the preceding scenario, we explored the case in which CSP takes the counteraction, which is going to have some latency between recognizing the raised alarm and responding to it. However, this is not the limitation of the proposed defense, and this latency overhead can be minimized by configuring the TDC sensor alarm output as a clock gate signal for the tenant, pausing further operation until the CSP decides on the appropriate action. This also prevents the attackers from performing another attack, such as a fault injection or a complete power failure.

6 Future Directions

This work focuses on providing safety and ensuring the security of tenants, where vendor-provided solutions fall short. Now, we discuss further initiatives that can assist users of multi-tenant FPGAs.

6.1 Fair Scheduling

Recent high-capacity FPGAs support multi-tenancy, allowing several tenants to utilize the same FPGA. This *sharing* of resources is done to maximize resource utilization and minimize service costs. Resource scheduling is critical to enabling efficient, faster, and fairer multitenancy. Traditional OS scheduling methodologies cannot seamlessly migrate to cloud FPGA virtualization systems due to the unique architecture of FPGAs in comparison to other processing systems (such as CPUs and GPUs).

From a brief overview of existing efforts, we believe a research gap exists regarding fair and efficient resource utilization. A quantitative study that evaluates technological limits and provides new scheduling algorithm metrics seems promising. Multi-tenant FPGA scheduling algorithms will ensure safety and reliability by considering energy, delay, and fairness. A fair scheduling method will also optimize resource use throughout the tenancy cycle.

6.2 Secure Tenant's Task Preemption

Task preemption refers to the system's ability to stop or pause a current task in favor of a higher-priority task. Preemption is a well-studied concept in the domain of operating systems (OS) [89]. However, the heterogenous nature of FPGA's underlying resources does not support the *traditional* preemption mechanisms proposed for embedded devices such as microcontrollers [90, 91], GPUs [92, 93], CPUs [94, 95], etc. As a result, there is a significant gap between theoretical and practically achievable support for preemption in multi-tenant FPGAs.

The state of each heterogeneous FPGA resource must be read out, saved, and restored carefully while keeping track of FPGA's clock cycles and practical limitations in mind. A well-thought procedure that pauses the system clock at specific intervals must be designed so that the design's state involving multicycle and clock-domain crossover is not harmed. Moreover, there are also information security principles to consider while enabling preemption. The *confidentiality* of the tenant's data must be maintained when its state is read out and saved. The *integrity* of the tenant's data state must remain intact while it remains suspended in a hibernating state. Likewise, the suspended state must also stay accessible to ensure that activities may be finished on schedule and not become a bottleneck for *availability*. Therefore, task preemption must also be *secure*. Due to the scarcity of vendor support in this area, we believe that a plug-and-play solution offering *secure preemption* will drastically expand the capabilities of multi-tenant FPGAs.

6.3 Standardized Open-Source Benchmarking

Benchmarks are essential in testing the efficacy of any novel design, algorithm, or system. Adequate benchmarks help replicate the challenges a system/application may face in the real world. They also help to capture the subtleties of a system or to identify corner cases that may have gone unnoticed during system development. Unfortunately, present multi-tenant FPGAs lack a framework and a benchmark suite upon which the research community and academia have a consensus. Researchers and industry are either developing their *application tailored* benchmarks [96–100] in order to gauge their developed systems or using benchmarks that were standardized with different philosophies and goals in mind [101–104].

The present literature lacks a *standardized* work that captures the core demands of multi-tenant FPGAs and the

issues associated with creating, deploying, and administering applications on such FPGAs. This presents an appealing opportunity to be taken advantage of by the talented minds working in this research line. Standardization, open-source accessibility, and formal verification must all be considered when determining the work's success.

6.4 Rescuing Multi-tenant FPGAs using ML

ML models consist of multiple layers that are trained over a long period of time and are crucial piece of intellectual property. A lot of these models are trained on the cloud to speed up the process and can be considered a major consumer of multi-tenant FPGAs. By leveraging remote sensors [50], an attacker can easily extract the critical parameters involved in these models. Address-redirection and task-hiding attacks are a new class of threats that are aimed at ML models [105].

In address-redirection, a malicious tenant can divert bitstream loading from unauthorized memory locations. As a result, attackers can use illegal hardware tasks to redirect communications between cloud applications and hardware tasks to steal critical data. Task-hiding exploits reconfigure the FPGA with a malicious bitstream by circumventing reconfiguration management. This method resembles processor kernel rootkit concealment.

ML tools and techniques should be used to develop defenses to prevent such attacks. As real-world threats develop, a defense that uses deep learning and machine learning will be vital.

7 Conclusions

This paper examined the various deployment strategies beneficial in securing FPGA-based cloud computing, highlighting various adversary models and associated security vulnerabilities. In particular, we explored the FaaS model and analyzed whether or not it conformed to the guidelines of the CIA triad. Through experimentation, we exposed the vulnerabilities in multi-tenant FPGAs that an attacker can exploit to extract the weights of an ML model by launching a remote side-channel attack. Our experiments demonstrate that existing academic proposals and vendor-provided tools are insufficient and inefficient for maintaining the *integrity* of tenants. The prior work exerts increased latency and overhead while only being configurable at *design time*. Our proposed solution meets the needs of modern users renting multi-tenant FPGAs with minimal overhead. We addressed the drawbacks in the prior works while maintaining low overhead and provided a fine-grained option that is configurable at *run-time*.

The proposed methodologies can help maintain security and safety by providing an improved secure interconnect infrastructure for multi-tenant FPGAs. This paper informs cloud service providers and warns end-users regarding the potential issues of the PR. Our results indicate that a knowledgeable adversary can exploit sudden voltage drops of PR operation to launch a DoS attack. The proposed RTL-based defense mechanism has the potential to identify attacks targeting the *availability* of multi-tenant FPGAs while being generic and scalable. The study advises system designers and hardware engineers about needing more research in this domain to guarantee that multi-tenant FPGAs and cloud service providers follow the CIA paradigm.

Statements and Declarations

Funding This work was funded through the Office of Naval Research (ONR) grant N00014-21-1-2809. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Conflict of Interest The authors declare no competing interests. The work is done by NCSU and supported by ONR.

Author Contribution All authors contributed to the design and implementation of the proposed evaluation methodology. All authors participated in the writing and revision of the paper. All authors have read and approved the submitted manuscript.

Availability of Data and Materials These declarations are "not applicable."

Ethical Approval Not applicable.

References

1. Xilinx (2022) Vivado design suite UG: dynamic function exchange (UG909). https://docs.amd.com/v/u/aKelve5HpAzsEgiAsRfyQA
2. Cloud A (2018) Deep dive into Alibaba cloud F3 FPGA as a service instances. Retrieved November 26:2020
3. Amazon (2019) Amazon AWS: Amazon EC2 F1 (2019) Amazon AWS. https://aws.amazon.com/ec2/instance-types/f1/
4. Xing J, Liu Y, Ge S, Li Y, Meng J (2019) An FPGA-based HF/VHF/UHF integrated self-interference cancellation system. 2019 Joint International Symposium on Electromagnetic Compatibility. Sapporo and Asia-Pacific International Symposium on Electromagnetic Compatibility (EMC Sapporo/APEMC), IEEE, pp 725–728
5. Module, Cobham AES Cryptographic Firmware-Hybrid (2015) Cobham TCS limited. https://seccerts.org/fips/f40ddf6742b7761b/target.pdf
6. Naghibijouybari H, Neupane A, Qian Z, Abu-Ghazaleh N (2018) Rendered insecure: Gpu side channel attacks are practical. Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. pp 2139–2153
7. Dubey A, Karabulut E, Awad A, Aysu A (2022) High-fidelity model extraction attacks via remote power monitors. 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS). IEEE, pp 328–331

8. Mahmoud DG, Lenders V, Stojilović M (2022) Electrical-level attacks on CPUs, FPGAs, and GPUs: survey and implications in the heterogeneous era. ACM Comput Surv (CSUR) 55(3):1–40
9. Karabulut E, Yuvarajappa C, Shaik MI, Potlurix S, Awad A, Aysu A (2022) Pr crisis: Analyzing and fixing partial reconfiguration in multi-tenant cloud fpgas. Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security. pp 101–106
10. Restuccia F, Meza A, Kastner R (2021) AKER: A design and verification framework for safe and secure soc access control. 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD). IEEE, pp 1–9
11. Xilinx Inc (2022) PG059 AXI interconnect product guide. v2.1, PG059. https://docs.amd.com/r/en-US/pg059-axi-interconnect
12. Vipin K, Fahmy SA (2014) ZyCAP: efficient partial reconfiguration management on the Xilinx Zynq. IEEE Embed Syst Lett 6(3):41–44. IEEE
13. Sultana B, Ullah A, Malik AA, Zahir A, Reviriego P, Muslim FB, Ullah N, Ahmad W (2021) VR-ZYCAP: a versatile resourse-level ICAP controller for ZYNQ SOC. Electronics 10(8):99. MDPI
14. Vipin K, Fahmy SA (2012) A high speed open source controller for FPGA partial reconfiguration. 2012 International Conference on Field-Programmable Technology. IEEE, pp 61–66
15. Knodel O, Spallek RG (2015) RC3E: provision and management of reconfigurable hardware accelerators in a cloud environment. arXiv preprint http://arxiv.org/abs/1508.06843
16. Kao C (2005) Benefits of partial reconfiguration. Xcell Journal 55:65–67
17. Rihani MA, Nouvel F, Prévotet JC, Mroue M, Lorandel J, Mohanna Y (2016) Dynamic and partial reconfiguration power consumption runtime measurements analysis for ZYNQ SoC devices. 2016 International Symposium on Wireless Communication Systems (ISWCS). IEEE, pp 592–596
18. McDonald EJ (2008) Runtime FPGA partial reconfiguration. 2008 IEEE Aerospace Conference. IEEE, pp 1–7
19. Liu S, Pittman RN, Forin A, Gaudiot JL (2013) Achieving energy efficiency through runtime partial reconfiguration on reconfigurable systems. ACM T Embed Comput S (TECS) 12(3):593–660. ACM New York, NY, USA
20. Liu M, Kuehn W, Lu Z, Jantsch A (2009) Run-time partial reconfiguration speed investigation and architectural design space exploration. 2009 International Conference on Field Programmable Logic and Applications. IEEE, pp 498–502
21. Dubey A, Cammarota R, Aysu A (2020) BoMaNet: Boolean masking of an entire neural network. Proceedings of the 39th International Conference on Computer-Aided Design. pp 1–9
22. Moini S, Tian S, Holcomb D, Szefer J, Tessier R (2021) Remote power side-channel attacks on BNN accelerators in FPGAs. 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, pp 1639–1644
23. Karabulut E, Awad A, Aysu A (2023) SS-AXI: Secure and Safe Access Control Mechanism for Multi-Tenant Cloud FPGAs. 2023 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, pp 1–5
24. Xilinx User Guide 7 Series fpgas configuration. UG470, v1 11:1–176 11:1–176
25. Zynq X (2016) 7000 All Programmable SoC Overview. DS190 (v1. 10)0. (accessed on 2 Nov 2016)
26. AXI AAA, Specification-AXI AP (2011) Axi4, and axi4-lite, ace and ace-lite. tech rep, Technical report
27. Alam MM et al (2019) Ram-jam: remote temperature and voltage fault attack on FPGAs using memory collisions. In: Workshop on FDTC, IEEE, pp 48–55
28. Gnad DR, Oboril F, Tahoori MB (2017) Voltage drop-based fault attacks on FPGAs using valid bitstreams. 2017 27th International Conference on Field Programmable Logic and Applications (FPL). IEEE, pp 1–7
29. Krautter J, Gnad DR, Tahoori MB (2018) FPGAHammer: remote voltage fault attacks on shared FPGAs, suitable for DFA on AES. IACR Trans Cryptogr Hardw Embed Syst 44–68
30. Zhao M, Suh GE (2018) FPGA-based remote power side-channel attacks. 2018 IEEE Symposium on Security and Privacy (SP). IEEE, pp 229–244
31. Dessouky G, Sadeghi AR, Zeitouni S (2021) SoK: Secure FPGA multi-tenancy in the cloud: Challenges and opportunities. 2021 IEEE European Symposium on Security and Privacy (EuroS &P. IEEE, pp 487–506
32. Dastres R, Soori M (2020) Impact of meltdown and spectre on cpu manufacture security issues. Int J Eng Sci Technol 18(2):62–69
33. Mittal S, Vetter JS, Li D (2014) A survey of architectural approaches for managing embedded dram and non-volatile on-chip caches. IEEE Trans Parallel Distrib Syst 26(6):1524–1537
34. Mittal S, Inukonda MS (2018) A survey of techniques for improving error-resilience of dram. J Syst Archit 91:11–40. Elsevier
35. Corbett JD (2012) The Xilinx isolation design flow for fault-tolerant systems. Xilinx White Paper WP412
36. Malik AA, Ullah A et al (2020) Isolation design flow effectiveness evaluation methodology for Zynq SoCs. Electronics 9(5):814
37. Krautter J, Gnad DR, Schellenberg F, Moradi A, Tahoori MB (2019) Active fences against voltage-based side channels in multi-tenant FPGAs. 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, pp 1–8
38. Pham K, Horta E, Koch D, Vaishnav A, Kuhn T (2018) IPRDF: An isolated partial reconfiguration design flow for Xilinx FPGAs. 2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). IEEE, pp 36–43
39. Giechaskiel I, Rasmussen KB, Eguro K (2018) Leaky wires: Information leakage and covert communication between FPGA long wires. Proceedings of the 2018 on Asia Conference on Computer and Communications Security. pp 15–27
40. Luo Y, Xu X (2019) HILL: A hardware isolation framework against information leakage on multi-tenant FPGA long-wires. 2019 International Conference on Field-Programmable Technology (ICFPT). IEEE, pp 331–334
41. Yazdanshenas S, Betz V (2018) Improving confidentiality in virtualized FPGAs. 2018 International Conference on Field-Programmable Technology (FPT). IEEE, pp 258–261
42. Yazdanshenas S, Betz V (2019) The costs of confidentiality in virtualized FPGAs. IEEE Trans Very Large Scale Integr (VLSI) Syst 27(10):2272–2283. IEEE
43. Hori Y, Satoh A, Sakane H, Toda K (2008) Bitstream encryption and authentication with AES-GCM in dynamically reconfigurable systems}. 2008 International Conference on Field Programmable Logic and Applications. IEEE, pp 23–28
44. Schellenberg F, Gnad DR, Moradi A, Tahoori MB (2021) An inside job: remote power analysis attacks on FPGAs. IEEE Des Test 38(3):58–66
45. Gravellier J, Dutertre JM, Teglia Y, Loubet-Moundi P (2019) High-speed ring oscillator based sensors for remote side-channel attacks on FPGAs. In: 2019 International conference on ReConFigurable computing and FPGAs (ReConFig), IEEE, pp 1–8
46. Glamočanin O, Coulon L, Regazzoni F, Stojilović M (2020) Are cloud FPGAs really vulnerable to power analysis attacks? 2020 Design, Automation \& Test in Europe Conference & Exhibition (DATE). IEEE, pp 1007–1010
47. Donchez S, Wang X (2022) Memory Isolation for Multi-Tenant Data Integrity in Cloud MPSoC FPGAs. 2022 IEEE 13th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). IEEE, pp 0515–0521
48. Versal ACAP (n. d.) Technical reference manual. Xilinx, uM011

49. Zynq UltraScale (n. d.) MPSoC Technical Reference Manual, UG1085
50. Tian S, Moini S, Wolnikowski A, Holcomb D, Tessier R, Szefer J (2021) Remote power attacks on the versatile tensor accelerator in multi-tenant FPGAs. 2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, pp 242–246
51. Khawaja A, Landgraf J, Prakash R, Wei M, Schkufza E, Rossbach CJ (2018) Sharing, protection, and compatibility for reconfigurable fabric with AmorphOS. 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). pp 107–127
52. Korolija D, Roscoe T, Alonso G (2020) Do OS abstractions make sense on FPGAs? 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). pp 991–1010
53. Ma J, Zuo G, Loughlin K, Cheng X, Liu Y, Eneyew AM, Qi Z, Kasikci B (2020) A hypervisor for shared-memory FPGA platforms. Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. pp 827–844
54. Provelengios G, Holcomb D, Tessier R (2019) Characterizing power distribution attacks in multi-user FPGA environments. 2019 29th International Conference on Field Programmable Logic and Applications (FPL). IEEE, pp 194–201
55. Boutros A, Hall M, Papernot N, Betz V (2020) Neighbors from hell: Voltage attacks against deep learning accelerators on multi-tenant FPGAs. 2020 International Conference on Field-Programmable Technology (ICFPT). IEEE, pp 103–111
56. Guide GS (2010) Logicore™ IP system monitor wizard v2. Citeseer
57. Nguyen M, Tamburo R, Narasimhan S, Hoe JC (2019) Quantifying the benefits of dynamic partial reconfiguration for embedded vision applications. 2019 29th International Conference on Field Programmable Logic and Applications (FPL). IEEE, pp 129–135
58. Mahmoud D, Stojilović M (2019) Timing violation induced faults in multi-tenant FPGAs. 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, pp 1745–1750
59. Gravellier J, Dutertre JM, Teglia Y, Moundi PL, Olivier F (2020) Remote side-channel attacks on heterogeneous SoC. Smart Card Research and Advanced Applications: 18th International Conference, CARDIS 2019, Prague, Czech Republic, November 11--13, 2019, Revised Selected Papers 18. Springer, pp 109–125
60. Moini S (2023) Security of hardware accelerators in multi-tenant FPGA environments
61. Luo M, Suh GE (2021) Stealing zero-thresholding neural network data using timing channel
62. Hua W, Zhang Z, Suh GE (2022) Reverse-engineering CNN models using side-channel attacks. IEEE Des Test 39(4):15–22
63. Zhang Y, Yasaei R, Chen H, Li Z, Al Faruque MA (2021) Stealing neural network structure through remote FPGA side-channel analysis. IEEE Trans Inf Forensics Secur 16:4377–4388. IEEE
64. Güneysu T, Moradi A (2011) Generic side-channel countermeasures for reconfigurable devices. International Workshop on Cryptographic Hardware and Embedded Systems. Springer, pp 33–48
65. Danger JL, Guilley S, Bhasin S, Nassar M (2009) Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors. 2009 3rd International Conference on Signals, Circuits and Systems (SCS). IEEE, pp 1–8
66. Wild A, Moradi A, Güneysu T (2017) GliFreD: Glitch-free duplication towards power-equalized circuits on FPGAs. IEEE Trans Comput 67(3):375–87. IEEE
67. Crane S, Homescu A, Brunthaler S, Larsen P, Franz M (2015) Thwarting cache side-channel attacks through dynamic software diversity. NDSS. pp 8–11
68. Luo Y, Xu X (2020) A quantitative defense framework against power attacks on multi-tenant FPGA. Proceedings of the 39th international conference on computer-aided design. pp 1–9
69. McEvoy RP, Murphy CC, Marnane WP, Tunstall M (2009) Isolated WDDL: A hiding countermeasure for differential power analysis on FPGAs. ACM Transactions on Reconfigurable Technology and Systems (TRETS) 2(1):1–23. ACM New York, NY, USA
70. Ma J, Li X, Wang M (2014) Power-aware hiding method for s-box protection. Electron Lett 50(22):1604–1606
71. Gnad DRE, Krautter J, Tahoori MB, Schellenberg F, Moradi A (2020) Remote electrical-level security threats to multi-tenant FPGAs. IEEE Des Test. https://doi.org/10.1109/MDAT.2020.2968248
72. Guilley S et al (2008) Evaluation of power-constant dual-rail logic as a protection of cryptographic applications in FPGAs. In: Second International Conference on Secure System Integration and Reliability Improvement, pp 16–23
73. Quisquater JJ (2002) Side channel attacks. CRYPTREC Report. https://cir.nii.ac.jp/crid/1572824500017724160
74. Mirzargar SS, Stojilović M (2019) Physical side-channel attacks and covert communication on FPGAs: A survey. 2019 29th International Conference on Field Programmable Logic and Applications (FPL). IEEE, pp 202–210
75. Tiri K, Verbauwhede I (2004) Synthesis of secure FPGA implementations. Cryptology ePrint Archive
76. Yu P, Schaumont P (2007) Secure FPGA circuits using controlled placement and routing. Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis. pp 45–50
77. Moradi A, Standaert FX (2016) Moments-correlating DPA. Proceedings of the 2016 ACM Workshop on Theory of Implementation Security. pp 5–15
78. Knichel D, Moradi A, Müller N, Sasdrich P (2021) Automated generation of masked hardware. Cryptology ePrint Archive
79. Gao P, Xie H, Zhang J, Song F, Chen T (2019) Quantitative verification of masked arithmetic programs against side-channel attacks. International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, pp 155–173
80. Abromeit A, Bache F, Becker LA, Gourjon M, Güneysu T, Jorn S, Moradi A, Orlt M, Schellenberg F (2021) Automated masking of software implementations on industrial microcontrollers. 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, pp 1006–1011
81. Gnad D (2020) Remote attacks on FPGA hardware. Dissertation. Karlsruher Institut für Technologie (KIT), Karlsruhe
82. Benhani E, Bossuet L, Aubert A (2019) The security of arm trustzone in a FPGA-based SoC. IEEE Trans Comput 68(8):1238–1248
83. Factor M et al (2013) Secure logical isolation for multi-tenancy in cloud storage. 29th Symposium on Mass Storage Systems & Technologies. IEEE, pp 1–5
84. Hardy N (1988) The confused deputy: (or why capabilities might have been invented). ACM SIGOPS Operating Systems Review 22(4):36–38
85. Sun R, Qiu P, Lyu Y, Wang D, Dong J, Qu G (2021) Lightning: striking the secure isolation on GPU clouds with transient hardware faults. arXiv preprint http://arxiv.org/abs/2112.03662
86. La T, Pham K, Powell J, Koch D (2021) Denial-of-service on FPGA-based cloud infrastructures–attack and defense. IACR Transactions on Cryptographic Hardware and Embedded Systems. pp 441–464
87. La TM et al (2020) FPGADefender: malicious self-oscillator scanning for Xilinx UltraScale+ FPGAs. ACM Transactions on Reconfigurable Technology and Systems 13(3):1–31

88. Xilinx Inc (2021) PG267 AXI verification IP LogiCORE IP product guide. PG267
89. Farooq MO, Kunz T (2011) Operating systems for wireless sensor networks: a survey. Sensors 11(6):5900–5930
90. Hambarde P, Varma R, Jha S (2014) The survey of real time operating system: RTOS. 2014 International Conference on Electronic Systems. Signal Processing and Computing Technologies, IEEE, pp 34–39
91. Sabri C, Kriaa L, Azzouz SL (2017) Comparison of IoT constrained devices operating systems: A survey. 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA). IEEE, pp 369–375
92. Hong CH et al (2017) GPU virtualization and scheduling methods: a comprehensive survey. ACM Comput Surv(CSUR) 50(3):1–37
93. Yu F, Wang D, Shangguan L, Zhang M, Liu C, Chen X (2022) A survey of multi-tenant deep learning inference on GPU. arXiv preprint http://arxiv.org/abs/2203.09040
94. Salot P (2013) A survey of various scheduling algorithm in cloud computing environment. Int J Res Eng Technol 2(2):131–135
95. Chandiramani K, Verma R, Sivagami M (2019) A modified priority preemptive algorithm for CPU scheduling. Procedia Comput Sci 165:363–369. Elsevier
96. Reagen B, Adolf R, Shao YS, Wei GY, Brooks D (2014) MachSuite: benchmarks for accelerator design and customized architectures. In: IEEE International Symposium on Workload Characterization (ISWC), pp 110–119
97. Hara Y, Tomiyama H, Honda S, Takada H, Ishii K (2008) Chstone: A benchmark program suite for practical c-based high-level synthesis. 2008 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, pp 1192–1195
98. Zhou Y, Gupta U, Dai S, Zhao R, Srivastava N, Jin H, Featherston J, Lai YH, Liu G, Velasquez GA (2018) Rosetta: A realistic high-level synthesis benchmark suite for software programmable FPGAs. Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. pp 269–278
99. Gupta U, Dai S, Zhang Z (2015) Rosetta: A realistic benchmark suite for software programmable fpgas. Suite of Embedded Applications and Kernels Workshop (SEAK)
100. Goswami P, Shahshahani M, Bhatia D (2022) Mlsbench: A benchmark set for machine learning based fpga hls design flows. 2022 IEEE 13th Latin America Symposium on Circuits and System (LASCAS). IEEE, pp 1–4
101. Jamieson P, Rose J (2005) A verilog RTL synthesis tool for heterogeneous FPGAs. International Conference on Field Programmable Logic and Applications, 2005. IEEE, pp 305–310
102. Guo X, Dutta RG, He J, Tehranipoor MM, Jin Y (2019) Qif-verilog: Quantitative information-flow based hardware description languages for pre-silicon security assessment. 2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, pp 91–100
103. Hansen MC et al (1999) Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. Des Test Comput 16(3):72–80
104. Das SR, Mukherjee S, Petriu EM, Assaf MH, Sahinoglu M, Jone WB (2006) An improved fault simulation approach based on verilog with application to ISCAS benchmark circuits. 2006 IEEE Instrumentation and Measurement Technology Conference Proceedings. IEEE, pp 1902–1907
105. Elnaggar R, Karri R, Chakrabarty K (2019) Multi-tenant FPGA-based reconfigurable systems: attacks and defenses. 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, pp 7–12