A Cautionary Note on Building Multi-tenant Cloud-FPGA as a Secure Infrastructure

Yukui Luo, Yuheng Zhang, Shijin Duan, and Xiaolin Xu
Department of Electrical and Computer Engineering
Northeastern University, Boston, MA, USA

2022 International Conference on Field-Programmable Technology (ICFPT) | 978-1-6654-5336-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICFPT56656.2022.9974230

***Abstract*—Security concerns have been raised for multi-tenant cloud-FPGA in many recent works. While these existing works focused on studying the security of diverse cloud-FPGA applications, such as Advanced Encryption Standard (AES), the vulnerabilities associated with the inherent FPGA components are so far under-explored. For the first time, we investigate the robustness of a commonly used communication protocol for data exchange, Advanced eXtensible Interface (AXI), against fault injection attacks in a multi-tenant cloud-FPGA environment. We build an experimental setup with a commodity FPGA development kit and launch fault injection attacks on the shared power distribution network (PDN). To study the in-depth effects of such attacks, we characterize the voltage glitches of different attack patterns in a non-invasive manner, i.e., using electron magnetic measurement. We also mimic the real-world data transmissions using two crafted datasets with different statistical characteristics. The experimental results demonstrate the unique security vulnerabilities of the current AXI protocol in the context of a multi-tenant cloud-FPGA. Last, we discuss potential defense strategies against these vulnerabilities.**

***Index Terms*—Security, Cloud-FPGA, Fault Injection, Communication Protocol, Memory**

I. Introduction

Field-programmable Gate Arrays (FPGA) based accelerator has become an emerging solution for compute-intensive applications. In recent years, virtualization techniques have been proposed to fully exploit the flexibility and computing efficiency of FPGA hardware. For example, Khawaja *et al.* [14] propose a scheme that enables multiple users to share an FPGA chip. Although supporting higher resource utilization and flexibility, the multi-tenant cloud-FPGAs also faces unique security challenges, e.g., a malicious cloud-FPGA user can use the co-tenancy to snoop the sensitive information or inject faults into the applications of other users. As an example, Giechaskiel *et al.* [11] demonstrate a crosstalk-based side-channel on the FPGA long-wires, with which a malicious FPGA user can deduce the secret information transmitted over an adjacent long-wire. Similarly, the voltage fluctuation on the power distribution network (PDN) is demonstrated as a such side-channel, with which the adversary can extract the secret of a victim application like a neural network model architecture [25]. Besides, fault injection is found as another threat against multi-tenant cloud-FPGA security, in which the PDN of an FPGA chip can be manipulated to inject faults into the applications [10], [16].

While most existing works focus on investigating the security of diverse cloud-FPGA applications, few of them consider an important question: *is the current FPGA hardware architecture suitable for a multi-tenant cloud context*, i.e., are there any vulnerabilities associated with the FPGA hardware resources. Intuitively, like any circuit applications, these internal FPGA components are also vulnerable to similar attacks. Moreover, the reliability of inherent FPGA components (e.g., data transmission interface) plays an important role in the security and reliability of any FPGA application. For example, Rakin *et al.* [24] demonstrate that the data transmission channel between the on-chip and off-chip memories, if manipulated, can inject faults into the on-chip applications. Using deep neural network (DNN) models as the victim applications, they demonstrate that properly injected faults will make a DNN model inference accuracy close to random guess. Since the cloud-FPGAs are mainly built for compute-intensive tasks like DNN acceleration, their hardware implementations are usually configured to maximize the circuit execution speed or data throughput, and thus are highly vulnerable to fault injection attacks. However, without corresponding specific protection scheme for these inherent FPGA components, it is challenging to detect and prevent these attacks at run-time.

This paper, for the first time, investigates the impact of PDN-induced fault injection attacks on the data transmission interface of the multi-tenant cloud-FPGA. Without loss of generality, we adopt the Advanced eXtensible Interface (AXI) as a case study for the following reasons. As an emerging communication protocol in modern FPGAs, AXI provides high-speed and high-bandwidth interconnection between different IPs, micro-controllers, and kernels [6] on single-user FPGAs. These facts motivate us to rethink the security of these commonly used protocols for new infrastructures like the multi-tenant cloud-FPGA, such as, should we assume that these inherent FPGA components are still secure by default?

The contributions of this work are summarized as follows:

- For the first time, we investigate the inherent vulnerabilities associated with FPGA components in the context of multi-tenant cloud-FPGA. We build a prototype platform and conduct fine-grained and comprehensive fault injection attacks to study such unique vulnerabilities.
- We report that like any FPGA application, the communication protocol (or data transmission interface) of an FPGA is also vulnerable to fault injection attacks but with unique characteristics. We conduct analysis on the attack results and provide unique statistical patterns.
- We discuss the discovered vulnerabilities and envision the

potential defense strategies. Specifically, we review the existing defense solutions from recent works and discuss their applicability to protect the inherent FPGA modules.

II. Background and Related Work

A. Threat Model and Related Attacks

Without loss of generality, we adopt the representative threat model of multi-tenant cloud-FPGA in other recent works [11], [25]. In this threat model, the adversarial and victim users share the hardware resources of a cloud-FPGA, and they can execute their circuit applications simultaneously and independently. Specifically, the adversary can aggressively overload the power supply of an FPGA using power-hungry circuits, such as ring-oscillator (RO), to introduce transient voltage glitches in the power distribution network (PDN). As a result, the victim's FPGA applications will become unstable and execute incorrectly. Such PDN-induced fault injection attack has been validated in both Intel and Xilinx FPGAs [12], [22] to compromise different FPGA applications. For example, Mahmoud *et al.* [21] used the PDN-based fault injection to corrupt the entropy of the true random number generator. Boutros *et al.* [10] and Rakin *et al.* [24] used the PDN-induced fault to manipulate the accuracy of a DNN model inference.

These many related works, although demonstrating successful attacks against diverse FPGA applications, have not explored the security of FPGA hardware modules under such PDN-induced power attacks. Practically, a circuit application in multi-tenant cloud-FPGA may employ many auxiliary hardware components, such as multiple communication protocols like standard AXI, AXI-Lite, AXI-Stream, first-in-first-out (FIFO), and other simple direct connections. Although it is visible that all circuitry components on the FPGA will have vulnerabilities to such power attacks, there is no work that has explored and quantified such security issues. To fill this research gap, we use the standard AXI and AXI-Lite protocol as a case study in this work. The reason that we choose these communication protocols is because they are widely used inherent components in modern FPGAs, which are developed to connect user IP and external hardware components with high universality and efficiency. For example, the AXI protocols are adopted as a interface standard in the state-of-the-art Xilinx Versal FPGA's adaptable Network-on-Chip (NoC).

B. AXI Communication Protocol

As an interface designated for high-frequency and high-bandwidth systems [6], the AXI protocol is adopted by Xilinx to bridge the interconnection between different IP modules [3], [5]. AXI protocol uses AXI manager (AXI-M) and subordinate (AXI-S) interfaces to support the data or instruction exchange between the processing engine (PE) and external memory to enable high-speed FPGA development. There are five channels between AXI-M and AXI-S, namely write/read address (AW/AR) channels, write/read (W/R) data channels, and write response (B) channel [6]. In detail, the data transmission over the AXI protocol is initiated by the AW/AR channel. The typical handshaking signals for the transmitter (i.e., source) and receiver of the AXI include *VALID* and *READY* [6]. For example, the source (manager/subordinate) side generates a "VALID" signal to inform the receiver (subordinate/manager) side if the data is ready. Then, the receiver side starts to fetch the data and uses a "READY" signal to acknowledge the source for the next burst. In the W/R data channels, a "LAST" signal denotes the end of one burst transmission.

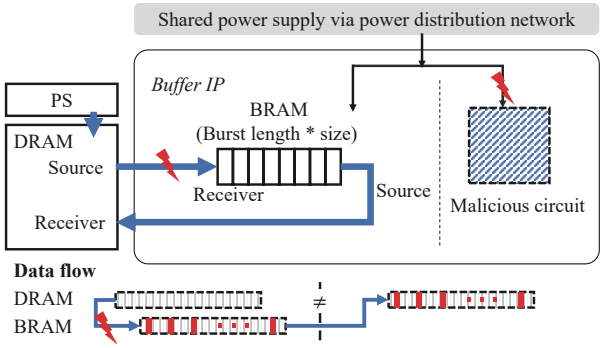


Fig. 1: Experimental setup overview.

III. PDN-induced Power Attack on AXI

To closely investigate the impact of injected fault by power attacks on AXI, we build a multi-tenant FPGA prototype on an Ultra96-V2 board [9], which has an Ultrascale + chip–ZU3EG and 2GB DDR4 on-board memory. We choose this evaluation kit since it integrates all common SoC-FPGA components, such as an SRAM FPGA, an ARM processor, DRAM controller/buses, and peripheral I/Os. Specifically, the FPGA device we used is built on TSMC 16nm low-power FinFET process, and its high-end version–VU9P is widely deployed in AWS F1 [2] and Alibaba Cloud F3 [1]. Therefore, this experimental setup enables us to: (1) study the impact of power attacks on the state-of-the-art FPGA fabrication technology node; (2) have a physically accessible platform to conduct a fine-grained and non-invasive investigation; (3) emulate the setup of a practical cloud-FPGA.

We use PYNQ [7] to bridge the FPGA evaluation board with the host PC and use Python programs to control the data flow between FPGA and the on-board memory. We analyze practical compute-intensive FPGA applications and find that most of them use the on-chip communication protocol to exchange a large amount of data between the on-chip and off-chip memories. For example, the deep learning implementations on FPGAs utilize *buffer IP*s to load the input data and model weight parameters for the processing engine (PE). We emulate these real-world setups, using on-chip BRAM to serve as the *buffer IP* to handle the data loading process.

A. Experimental System Overview

We show the experimental system in Fig. 1, which is built on a generic AXI-based data transmission channel. Specifically, the processing system (PS) side utilizes the DRAM controller to enable direct memory access (DMA), and the BRAM-based *buffer IP* uses AXI protocol to communicate with the off-chip

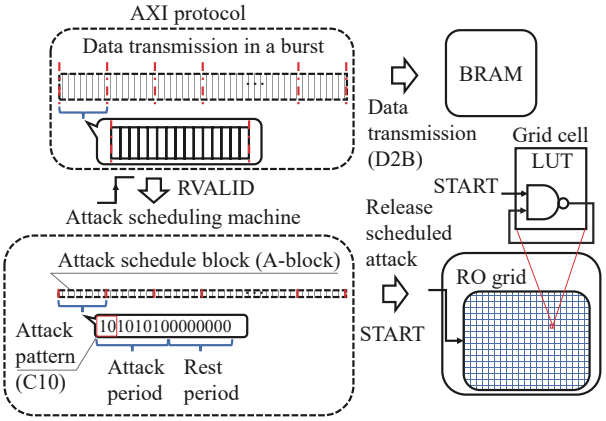


Fig. 2: Attacking diagram overview. An attack scheduling machine is used to trigger attacks on the data transmission.

memory (i.e., DRAM). The experimental setup handles two data transmission processes: DRAM to BRAM (D2B) data reading process and BRAM to DRAM (B2D) data written-back process. As illustrated in the bottom of Fig. 1, our experiment launches attacks during the D2B data transmission process, i.e., the data received by the BRAM will include faulty values. To facilitate quantitative attack analysis, we control the BRAM to write back these data to DRAM for comparison using the B2D channel, from which we can easily tell the difference between the original and fault injected data. We further build an *attacking scheduling machine* to enable fine-grained control of the malicious circuit, as detailed below.

1) Attack scheduling machine.: The AXI data transmission protocol is burst-based [4], i.e., the source side data is not always valid during transmission. Thus, it is meaningless to launch attacks if the transmission is idle for analysis purposes. To mitigate this issue, we synchronize the PDN-induced attacks with the data transmission using an *attack scheduling machine* (Fig. 2), to disable the malicious circuit (RO grid) if the source data is invalid. In addition, we simplify the triggering system by applying the data read-valid (RVALID) signal in the AXI-S. This signal can enforce the attack to be only triggered during the valid D2B data transmission. Leveraging this setup, we can learn how the PDN-induced fault injection affects the data transmission over AXI, i.e., by simply comparing the original DRAM data with the data written back from BRAM, as illustrated in the bottom of Fig. 1.

To facilitate fine-grained attacks for quantitative analysis, we execute the attack scheduling machine at the same clock frequency as the AXI interface. Specifically, we divide a single burst transmission into several attack blocks (*A-block* in Fig. 2), and each block is composed of "attack period" and "rest period". The "attack period" consists of different attack patterns such as "1", "10", and "100", in which "1" and "0" can enable and disable the malicious circuit, respectively. Fig. 2 illustrates a consecutive (C) attack pattern "10", denoted as "C10". The "Rest period" is composed of consecutive "0"s, which controls the time period without any attacks. Since the *attack scheduling machine* is synchronized with AXI-based data transmission, every single bit in the *A-block* determines whether a corresponding data transmission is under attack ("1") or not ("0").

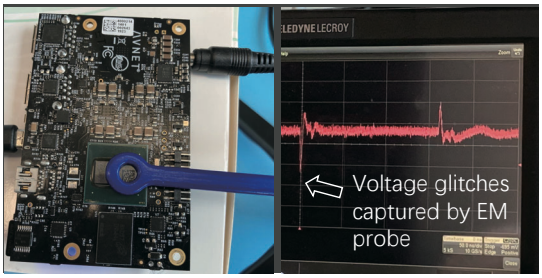
2) Malicious circuit.: Without loss of generality, we follow other related works to launch the PDN-induced fault injection on the AXI communication protocol [12], [13], [16], [17], [23]. Specifically, we instantiate a number of ring-oscillator (RO) as the malicious circuit, namely *RO grid* that shares the FPGA with the AXI data transmission setup, as shown in Fig. 1. Note that there are also other design options for the malicious circuit to satisfy different requirements, e.g., bypassing the checking of cloud deployment [17]. Since this paper focuses on investigating the FPGA communication protocols against PDN-induced power attacks, exploring novel malicious circuits is not our focus. In practice, a malicious cloud-FPGA user will explore different strategies to guide the PDN-induced fault injection attacks, i.e., random or side-channel based [12]. In our design, these ROs are jointly controlled by a "START" signal that is provided by the *attack scheduling machine*, as illustrated in Fig. 2.

B. Testing Data Generation

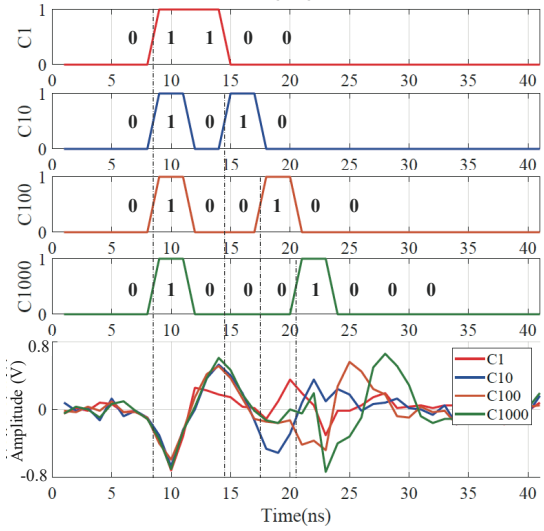
As a case study, we set the burst size of the *buffer IP* as 32-bit and the burst length as 256 (each burst has 256 data transmissions and each transmission handles 32-bit = 4-Byte data), and execute the IP buffer at 300MHz. We mimic the data transmission of real-world FPGA applications using two 15MB datasets with different statistical characteristics. Specifically, the first is a "random" dataset, with data values of 32-bit floating-point format and following uniform distribution. The second dataset is generated with 8-bit Gray code, and each value is reused four times to formulate a 32-bit data, representing the transmission of more regulated data structures. Note that we do not set up 32-bit Gray code dataset, which has 2^{32} numbers and cannot be exhaustively tested. Instead, we adopt a wheel encoder to generate the testing dataset to make all data have the same possibility to be attacked, i.e., the traversal of Gray code data is cycled. Therefore, each dataset will consume 15,360 ($= 15MB/32bits/256$) transmissions to finish the D2B or B2D transaction.

C. PDN-induced Voltage Glitch Characterization

Before evaluating the impact of PDN-induced fault injection attacks, we first utilize a Lecroy oscilloscope [8] to capture the voltage glitches caused by different attacking patterns. Since it is infeasible to directly measure the internal voltage glitches on PDN without introducing noise and bias, we use an electromagnetic (EM) probe with a sampling rate of 10GHz, to trace the energy radiation of the FPGA chip in a non-invasive manner. The experimental setup is shown in Fig. 3a. Although the energy radiation captured by the EM probe may not fully represent the voltage fluctuation in the chip, it provides us a highly accurate trace for attack activation, as well as the strength and timing characteristics of different attack patterns.



(a) EM trace-based voltage glitch measurement.



(b) Voltage glitches of different attack patterns.

Fig. 3: PDN-induced voltage glitch characterization.

Our evaluation applies each attack pattern twice to observe the corresponding voltage glitches. For example, the "C10" represents the attack pattern "consecutive 10", and we repeatedly apply it twice, like using "1010". Similarly, we apply three other attack patterns "C1", "C100", and "C1000", and their timing diagrams are plotted in Fig. 3b. The corresponding EM trace of each attack pattern is shown at the bottom of Fig. 3b. From these measurements, we observe the following characteristics: (1) For each attack pattern, the first "1" can activate the malicious circuit and incur a significant voltage drop on the PDN, which is then quickly recovered to the normal level. (2) For these applied attack patterns, the voltage drop usually takes 3 to 4 ns to reach the lowest level (at around the 10^{th} ns). (3) The attack pattern with more consecutive "1s" (e.g., "C1"), does not incur more voltage drop than others but slows down the voltage recovery. Also, the time interval between each activation of the malicious circuit plays an important role in the PDN-induced power attacks. This is due to the power management circuit, which can quickly replenish the capacitor-based power bank when the malicious circuit is deactivated.

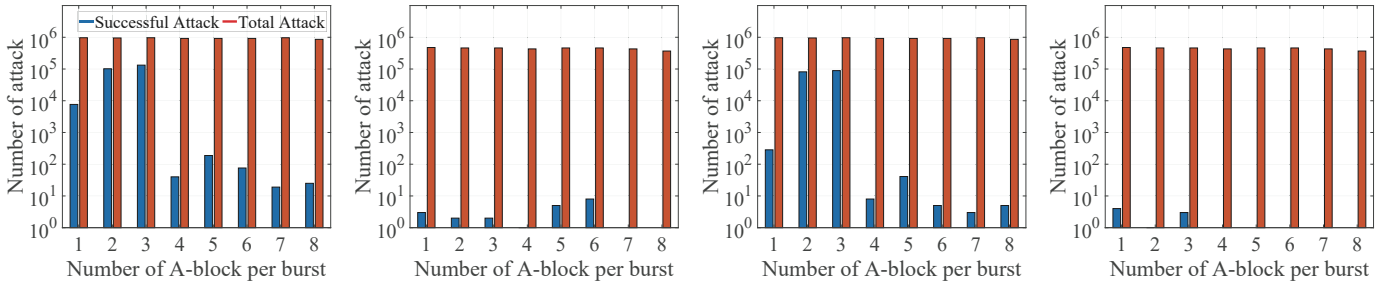
D. Fault Attack Result Analysis

a) ***Fault types****:* We observe two types of faults from the experimental results. We define the first type as *unrecoverable fault*, in which the transmitted data are received as incorrect on the receiver side, even after deactivating the RO grid. We find that such unrecoverable fault is caused by frequently triggering the malicious circuits within a short time duration, which possibly incurs system error or denial of service of the FPGA, as also reported in other works [12], [19]. We define the second type as *recoverable fault*, in which the receiver only receives incorrect values if the corresponding data transmission is attacked. Technically, a such recoverable fault is more stealthy since the adversary can accurately control the moments to inject faults, which is our focus in this paper. As an empirical configuration, we set the "rest period" (in Fig. 2) as 75% of the *A-block* size to avoid unrecoverable fault.

b) ***Attack results analysis****:* To conduct a comprehensive attack analysis, we apply *A-block*s with different "attack periods" to reconfigure the type and frequency of each attack pattern. For example, considering the data transmission has a burst length of 256 as indicated in Sec.III-B, then for an attack schedule with 4 *A-block*s, each *A-block* covers attacks on $256/4 = 64$ data transmissions. If adopting attack pattern "C10" and "attack period" equals to 25% of an *A-block* (to avoid the unrecoverable fault), the attack pattern "10" will be applied for 8 times. As a result, the *A-block* is 64h'AAAA_0000_0000_0000, in which each bit determines whether to attack a data transmission.

We change the number of *A-block*s from 1 to 8 to mimic different scenarios and plot the results in Fig. 4. We define an attack as successful if the integrity of corresponding data transmission is corrupted. We have two observations from the experimental results: (1) Comparing Fig. 4a and Fig. 4b (or Fig. 4c and 4d), we find attack pattern "C1" achieves higher attack success rate than "C10" in both datasets. (2) Comparing Fig. 4a and Fig. 4c (or Fig. 4b and Fig. 4d), we find the same attack pattern achieves similar attacking statistics on different datasets. Moreover, the dataset derived from Gray codes are more robust than the random dataset against the fault injection attacks. Here, we make an intuitive interpretation that Gray codes can tolerate more voltage-drop impact because only 1 bit is changed between each two adjacent data transmissions, which may enhance the integrity of each other. In general, using attack pattern "C1" and 3 *A-block* achieves the highest attack success rate, i.e., 23% in the random dataset and still 14% in the Gray code dataset.

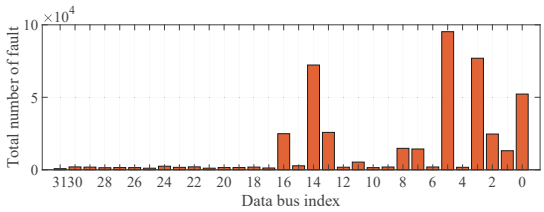
Fine-grained fault analysis: From Fig. 3b, we learn that *the root cause of PDN-induced fault injection* is a timing violation introduced by the voltage glitch directly that changes the data transmission speed (or propagation delay) of the victim circuits. For example, if a specific bit-line transmits 1-0-1 over three consecutive clock cycles, the PDN-induced fault could be injected on the second bit, which can be incorrectly sampled as the adjacent "1" by the receiver side. In contrast, if a bit-line transmits constant values (e.g., all-1), the PDN-induced fault injection is less likely to happen. These results are also in accordance with the fault injection-induced "data duplication" in [24], i.e., two similar and consecutive data packages are less likely to be injected with faults due to the high similarity



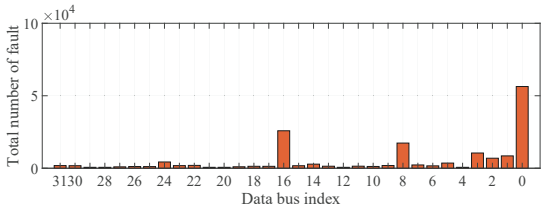
(a) Random dataset under "C1". (b) Random dataset under "C10". (c) Gray code dataset under "C1". (d) Gray code dataset under "C10".

Fig. 4: The fault injection results under different attack patterns.

that naturally enhances their integrity.



(a) Bit-wise fault in the random dataset



(b) Bit-wise fault in the Gray code dataset

Fig. 5: Fine-grained fault analysis over two testing datasets.

We use the experimental results with the highest attack success rate as a case study to explore the bit-level fault injected by the PDN-induced attacks on 32-bit data, and the results of both random and Gray code-based datasets are shown in Fig. 5a and Fig. 5b. From these results, we find that: (1) *Physical placement and routing of the victim impacts the fault injection results*. Considering that the values in the random dataset follow uniform distribution, thus each bit should have the same possibility of being corrupted. However, as shown in Fig. 5a, some specific bits are easier to be compromised in the random dataset. Therefore, we deduce the vulnerability difference should be caused by the physical placement and routing of these wires, i.e., the location and length of different wires make some of them more vulnerable to timing violations. This phenomenon is observed in the attacking result of the Gray code dataset as well, i.e., the lower bits are more vulnerable since we use the identical placement and routing configuration for both datasets. (2) *The data structure also plays an important role in the fault injection attack*. In Fig. 5b, each 8-bit data on the Gray code dataset shows a similar statistical distribution after fault injection, despite the attack success rate decreased along the data bit index. This is because we generate the 32-bit Gray code dataset by repeating each 8-bit data 4 times (see Sec. III-B). (3) *The flipping-frequency of a specific bit-index determines its vulnerability to fault injection attacks*. In Fig. 5b, we find the bit indexes 0, 8, 16, and 24 are easier to be attacked, due to the higher data flipping-frequencies in these indexes. For example, these bits closer to LSB are flipped more frequently across consecutive data transmissions. To further validate this, we cyclically shift the Gray code dataset and apply the same attacks, and find the indexes of vulnerable bits also shift accordingly with the data, which affirms our conclusion.

IV. Discussion

Multiple recent works have proposed protection schemes against secret-snooping or PDN-based fault injection attacks in a multi-tenant cloud-FPGA context. As a rough classification, they can be categorized as three different strategies:

Obfuscation strategy: Krautter *et al.* [15] proposed to obfuscate the side-channel leakages in a multi-tenant FPGA, which can effectively prohibit the application of correlation power analysis (CPA) of AES against the key extraction. Similarly, Luo *et al.* [18] proposed using obfuscation to mitigate the crosstalk-induced side-channel on FPGA long-wires. However, these strategies are either designated for specific security applications and against side-channel attacks, the faults can still be injected even after employing these obfuscation mechanisms.

Monitoring strategy: Some recent works [17], [20] proposed using on-chip sensors to monitor the voltage/delay fluctuation as a defense. However, from our exploration in Sec. III-D, the successful *recoverable fault* attack requires carefully crafted attack patterns, i.e., an appropriate rest period is very important in avoiding the shutdown and *unrecoverable fault*. From this perspective, these attacks inducing *recoverable fault* are more stealthy, and the their attacking frequency is low. Thus, the monitoring strategy may not capture such attacks, but could only be used as an indirect defense solution.

Dynamic frequency scaling strategy: Leveraging the monitoring scheme, Luo *et al.* [20] introduced a dynamic frequency scaling (DFS) framework, which can passively adjust the frequency of victim applications at run-time. Specifically, this method target executing a victim applications at a lower clock frequency, in order to improve its tolerance to the timing violations from fault injections. A critical issue of this method is the performance loss caused by frequency scaling/reduction.

Envisioned defense solution: As an emerging infrastructure designated for high-performance, the data exchange between the FPGA on-chip module (e.g., BRAM) and the external device (e.g., DRAM), rendering the data transmission interface or communication interface like AXI a tie-breaker of the entire computing infrastructure. Moreover, considering that most applications outsourced to cloud-FPGAs are compute-intensive, i.e., of high frequency and throughput, thus, the corresponding defense strategies should also be compatible with the high-performance communication protocols. We envision the following expected characteristics in the potential defense solutions, from a protocol-hardware co-design perspective, to secure data transmission interfaces like AXI without introducing high-performance loss. Note that these design requirements should be generally considered to any other FPGA components towards building a secure multi-tenant infrastructure.

From the protocol perspective, we should first consider the security of the controlling signals (e.g., data requests). Since if carrying faulty information, these controlling signals will directly incur faulty data transmission process or even unrecoverable faults like crash, as reported in our observation (Sec. III-D). Besides, due to the *unpredictable* behavior of an adversary, it is uncertain when and how the faults will be injected. Therefore, a generic defense strategy should consider employing integrity checking and error correction, e.g., using redundant transmission. For example, we can modify the handshaking protocol by sending redundant controlling signals, e.g., sending two data requests and comparing them to check the integrity. This strategy is feasible for the following reasons: (1) The data request frequency is usually lower than the data streaming; (2) The randomness of fault injection makes it less possible for the adversary to introduce exactly identical faults to consecutive data requests transmissions, even on the data requests of smaller sizes. Similarly, from the hardware perspective, we should prioritize the integrity protection of the controlling modules like the finite state machine, such as introducing self-monitoring mechanisms. Moreover, any defense strategy incurring a large hardware overhead should be adequately addressed. For example, we can protect the most security-critical and application-specific data transmission to balance the overhead and robustness.

V. Conclusion

This paper investigates the vulnerabilities associated with the internal hardware modules of an FPGA in the multi-tenant and cloud environment. Our study uses the commonly used AXI communication interface and PDN-induced power attacks as a case study. We conduct comprehensive and fine-grained analysis using different test datasets and attacking patterns, and the experimental results demonstrate unique vulnerabilities associated with the AXI protocol-based data transmission. Although this work uses AXI as a case study to explore the inherent vulnerabilities of FPGA hardware modules, its conclusion can be generally applied to other FPGA-specific hardware resources, highlighting the importance of exploring defense solutions for this emerging infrastructure.

References

[1] Create an f3 instance. https://www.alibabacloud.com/help/en/elastic-compute-service/latest/create-an-f3-instance.

[2] Developer preview – ec2 instances (f1) with programmable hardware. https://aws.amazon.com/cn/blogs/aws/developer-preview-ec2-instances-f1-with-programmable-hardware/.

[3] "Axi video direct memory access v6.3," 2017.

[4] "Vivado design suite axi referenc guide," 2017.

[5] "Axi block ram (bram) controller v4.1," 2019.

[6] "Introduction to amba axi," 2021, https://developer.arm.com/documentation/102202/latest/.

[7] "Pynq: Python productivity," 2021, http://www.pynq.io/.

[8] "Teledyne LeCroy - Oscilloscope," 2021, https://teledynelecroy.com/oscilloscope/.

[9] "Ultra96-v2 single board computer hardware user's guide," 2021, https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/ultra96-v2/.

[10] A. Boutros, M. Hall, N. Papernot, and V. Betz, "Neighbors from hell: Voltage attacks against deep learning accelerators on multi-tenant fpgas," in *2020 International Conference on Field-Programmable Technology*.

[11] I. Giechaskiel, K. B. Rasmussen, and K. Eguro, "Leaky wires: Information leakage and covert communication between fpga long wires," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, 2018, pp. 15–27.

[12] D. R. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on fpgas using valid bitstreams," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–7.

[13] C. Jin, V. Gohil, R. Karri, and J. Rajendran, "Security of cloud fpgas: A survey," *arXiv preprint arXiv:2005.04867*, 2020.

[14] A. Khawaja, J. Landgraf, R. Prakash, M. Wei, E. Schkufza, and C. J. Rossbach, "Sharing, protection, and compatibility for reconfigurable fabric with amorphos," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 107–127.

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

[16] J. Krautter, D. R. Gnad, and M. B. Tahoori, "Fpgahammer: remote voltage fault attacks on shared fpgas, suitable for dfa on aes," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018.

[17] T. M. La, K. Matas, N. Grunchevski, K. D. Pham, and D. Koch, "Fpgadefender: Malicious self-oscillator scanning for xilinx ultrascale+ fpgas," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 13, no. 3, pp. 1–31, 2020.

[18] Y. Luo, S. Duan, and X. Xu, "Fpgapro: A defense framework against crosstalk-induced secret leakage in fpga," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, pp. 1–31, 2021.

[19] Y. Luo, C. Gongye, S. Ren, Y. Fei, and X. Xu, "Stealthy-shutdown: Practical remote power attacks in multi-tenant fpgas," in *IEEE 38th International Conference on Computer Design (ICCD)*, 2020.

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

[21] D. Mahmoud and M. Stojilović, "Timing violation induced faults in multi-tenant fpgas," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1745–1750.

[22] G. Provelengios, D. Holcomb, and R. Tessier, "Characterizing power distribution attacks in multi-user fpga environments," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2019, pp. 194–201.

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

[24] A. S. Rakin, Y. Luo, X. Xu, and D. Fan, "Deep-dup: An adversarial weight duplication attack framework to crush deep neural network in multi-tenant fpga," *arXiv preprint arXiv:2011.03006*, 2020.

[25] S. Tian, S. Moini, A. Wolnikowski, D. Holcomb, R. Tessier, and J. Szefer, "Remote power attacks on the versatile tensor accelerator in multi-tenant fpgas," in *Proceedings of the International Symposium on Field-Programmable Custom Computing Machines*, May 2021.