

# FLARE: Fault Attack Leveraging Address Reconfiguration Exploits in Multi-Tenant FPGAs

Jayeeta Chaudhuri<sup>†</sup>, Hassan Nassar<sup>†</sup>, Dennis R.E. Gnad<sup>†</sup>, Jörg Henkel<sup>†</sup>,  
Mehdi B. Tahoori<sup>†</sup>, and Krishnendu Chakrabarty<sup>†</sup>

<sup>†</sup>School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ, USA

<sup>‡</sup>Institute of Computer Engineering, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

**Abstract**—Modern FPGAs are increasingly supporting multi-tenancy to enable dynamic reconfiguration of user modules. While multi-tenant FPGAs improve utilization and flexibility, this paradigm introduces critical security threats. In this paper, we present FLARE, a fault attack that exploits vulnerabilities in the partial reconfiguration process, specifically while a user bitstream is being uploaded to the FPGA by a reconfiguration manager. Unlike traditional fault attacks that operate during module runtime, FLARE injects faults in the bitstream during its reconfiguration, altering the configuration address and redirecting it to unintended partial reconfigurable regions (PRRs). This enables the overwriting of pre-configured co-tenant modules, disrupting their functionality. FLARE leverages power-wasters that activate briefly during the reconfiguration process, making the attack stealthy and more challenging to detect with existing countermeasures. Experimental results on a Xilinx Pynq FPGA demonstrate the effectiveness of FLARE in compromising multiple user bitstreams during the reconfiguration process.

## I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are increasingly being used as customized accelerator platforms to support high-performance computing and accelerating compute-intensive workloads. Major cloud service providers (CSPs) such as Amazon, Alibaba Cloud, and Microsoft offer FPGA-accelerated instances, which allow users to deploy their custom modules with high flexibility and efficiency, particularly for artificial intelligence and machine learning applications [1], [2]. With their scalability across several compute-intensive tasks, FPGAs are being increasingly adopted for virtualization and multi-tenancy applications. The multi-tenant approach allows CSPs to maximize resource utilization by allowing users access to simultaneously reconfigure their applications on separate modules of the same FPGA.

However, the multi-tenancy approach introduces several security threats, particularly physical and remote attacks. Physical attacks occur when an attacker gains direct access to the FPGA system, enabling them to measure power consumption or measure electrical current [3]–[6]. However, emerging threats demonstrate that fault attacks can be activated remotely via untrusted users using on-chip sensors that impact the functionality of co-tenant modules, without requiring physical access to the FPGA [7]. In [8], ring oscillator (RO)-based circuits have been used to create excessive voltage fluctuations, that lead to a permanent crash of the FPGA. Building on this, [9]–[11] showed that by repeatedly toggling a set of ROs, attackers can induce fault attacks on an AES module. In [12], certain toggling frequencies of an RO grid were shown

to induce faults during bitstream reconfiguration. Faults are injected into the bitstream by targeting the reconfiguration manager (RM), which uploads and configures bitstreams in the partial reconfigurable regions (PRRs) of the FPGA. Unlike [8]–[11], these faults persist even after the attack ceases, resulting in computational errors after FPGA deployment.

Prior runtime fault attacks [8]–[11] require extended attack durations, as power-wasters need to remain activated throughout the operation of the tenant modules. This requirement increases the likelihood of attack detection by voltage-monitoring schemes [13], [14]. In [12], the attack duration is reduced by activating power-wasters only during partial reconfiguration. However, [12] is successful only when Cyclic Redundancy Check (CRC) is disabled, which makes the attack more detectable. Moreover, all the aforementioned attacks are limited to disrupting only a single tenant module.

In this paper, we introduce FLARE, a novel fault attack that adversely impacts multiple tenants by altering the reconfiguration address of a user bitstream rather than the bitstream data. To achieve this, we employ malicious power-wasters that induce faults in the bitstream while it is being configured onto the FPGA through the RM. These faults alter the ‘select’ part of the bitstream, which specifies the configuration address that determines the exact PRR where the bitstream is to be loaded [13]. By injecting faults into the configuration address, the bitstream is incorrectly redirected, resulting in the overwriting of PRRs of other co-tenants. This leads to erroneous computation and potential Denial-of-Service (DoS) for the user bitstream. The ‘select’ part of the bitstream is much smaller than the full bitstream, consisting of only a few configuration words. By precisely targeting fault injection in the ‘select’ part, the attack duration is significantly minimized, making it *stealthy* and harder to detect by the schemes described in [13], [14]. In our paper, a stealthy attack is defined as the attacker’s ability to evade detection by countermeasures that rely on monitoring extended fault injection activity.

We demonstrate the effectiveness of FLARE using benign design modules that are configured within a multi-tenant FPGA setup. Fig. 1 illustrates the proposed fault attack setup.

The key contributions of this paper are as follows.

- We determine the appropriate configuration parameters of several power-wasters to induce a successful fault attack.
- We demonstrate the first address manipulation fault attack that targets the process of FPGA partial reconfiguration.
- We show that targeted fault injections in the configuration

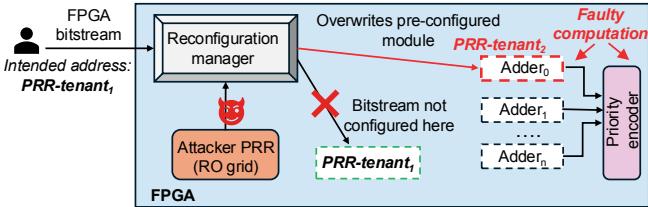


Fig. 1: Proposed attack setup of FLARE (Adder<sub>0</sub>: benign module configured in address *PRR-tenant*<sub>2</sub>. Due to fault-injection, an user bitstream is redirected to *PRR-tenant*<sub>2</sub> instead of its intended address *PRR-tenant*<sub>1</sub>).

address section of the bitstream can disrupt the functionality of other modules sharing the FPGA fabric.

- We evaluate FLARE on several modules, including an AES cryptography implementation and adder instances.
- We implement FLARE on hardware, targeting the Xilinx Pynq-Z1 FPGA board.

The remainder of the paper is organized as follows. Section II provides an overview of prior attacks on multi-tenant FPGAs and establishes the motivation of our attack. The threat model is described in Section III. Section IV presents the attack framework and the experimental setup. Section V provides experimental results. Finally, Section VI concludes the paper.

## II. RELATED PRIOR WORK AND MOTIVATION

### A. Prior Fault Attacks on Multi-Tenant FPGAs

In FPGAs, PRRs are designed to maintain logical isolation to ensure secure operation for co-tenant modules [15]. However, recent work explores the vulnerabilities of multi-tenant FPGAs to remote fault attacks. An adversary can deploy power-wasters on the FPGA to exploit the shared power distribution network (PDN), causing rapid voltage fluctuations.

In [8], authors implemented ROs for generating abrupt voltage drops and showed their effectiveness in DoS and crashing of the FPGA. Extending upon this approach, [9] explored the potential of ROs to induce voltage drop-based fault-injections in an AES module. By selecting the appropriate RO toggling frequency, faults can be injected in AES, resulting in incorrect ciphertext generation or key recovery. [16]–[18] explored non-combinational ROs and glitch amplification circuits that evade the Amazon Web Services design rule check (DRC). In [11], a runtime fault attack was demonstrated that did not require combinational loops or glitch-inducing designs. Instead, the attack exploited short circuits within the FPGA to force simultaneous writes to the same memory address in a dual-port RAM. These memory collisions caused permanent alterations in the bitstream configuration. In contrast to the aforementioned attacks that are executed at runtime, [12] explores a fault attack that is carried out during the process of bitstream reconfiguration. In this approach, faults are injected directly into the bitstream, which persist even after the bitstream is configured onto the FPGA.

### B. Fault Attack Countermeasures

Several countermeasures in the literature aim to detect fault attacks that employ power-wasters; these countermeasures are

TABLE I: Comparison of FLARE with prior fault attacks on multi-tenant FPGAs (Note: Few  $\mu$ s (ms) refers to a duration of a few microseconds (milliseconds)).

Attack method	Target module	Loop-free oscillator evaluated?	Attack duration	Address altered?	Tenant impacted
[11]	RAM	✗	Continuous	✗	Single
[9]	AES	✗	Continuous	✗	Single
[8]	FPGA	✗	Continuous	✗	Single
[10]	Bitstream	✗	Continuous	✗	Single
[12]	Bitstream	✗	Few ms	✗	Single
[17]	RSA	✗	Continuous	✗	Single
[18]	FPGA	✓	Continuous	✗	Single
<b>FLARE</b>	Bitstream, FPGA PRR(s)	✓	Few $\mu$ s	✓	Multiple

focused on identifying runtime fault attacks. In [13], fault attacks are detected using voltage drop-based sensors. This technique aims at disabling the interconnects of the attacker module, blocking it from FPGA configuration. Another fault-monitoring scheme, presented in [14], disables the clock signal of the fault-impacted tenant to halt the activity of power-wasters. These methods rely on power-wasters being continuously activated for a significantly long duration to detect fault attacks. Note that all prior runtime fault attacks [8]–[11] persist throughout the victim module’s operation. Such prolonged attack durations increase the likelihood of detection by the techniques presented in [13], [14].

In contrast, FLARE is stealthier and faster, with a higher likelihood of evading known countermeasures. FLARE minimizes the attack duration by precisely activating the power-wasters only when the bitstream configuration address is being loaded into the RM. Unlike [12], which is detected by CRC, FLARE evades CRC detection successfully. Table I presents a qualitative comparison of FLARE with prior fault attacks.

## III. THREAT MODEL

We consider a multi-tenant FPGA setup, i.e., multiple users (tenants) deploy their modules independently on different regions of the FPGA. In this scenario, an attacker is a malicious third-party user who reconfigures one of the PRRs of the FPGA with a malicious power-waster. The goal of the attacker is to manipulate the configuration address of the user bitstream while it is being loaded onto the RM. The RM, which is configured on the same FPGA, loads the bitstream in its memory and subsequently configures it onto the designated PRRs of the FPGA. However, the RM is not designed with security as a focus, making it vulnerable to exploitation.

**Attacker Knowledge:** The RM includes a status register that can be read by applications and tenants without requiring access privileges [19]. An attacker can monitor this status to detect when a bitstream is being loaded into the RM, including the critical timing of loading the configuration address.

**Attacker Capability:** The attacker activates the power-wasters when the configuration address of the bitstream is being loaded into the RM. Upon activation of the power-wasters, there are significant voltage fluctuations in the PDN, which induces faults in the bitstream.

**Attack Outcome:** The injected faults in the bitstream alter its configuration address, redirecting it to the wrong PRR.

Note that the RM is capable of loading both encrypted



Fig. 2: Structure of a Xilinx partial bitstream (the target region for attack is highlighted in red).

and unencrypted bitstreams. However, bitstreams are typically decrypted before FPGA configuration [20]. Therefore, in our experiments, we specifically focus on decrypted bitstreams.

#### IV. FAULT ATTACK TARGETING RECONFIGURATION ADDRESS MANIPULATION

In this section, we provide a detailed analysis of FLARE, which targets configuration address manipulation. First, we implement several power-wasters that have been shown to cause voltage-based fault attacks and DoS [8], [9], [18], [21]. We evaluate the appropriate configuration parameters of these power-wasters for triggering a successful fault attack on the FPGA. Next, we describe how activation of these power-wasters during the partial reconfiguration process of bitstreams can adversely impact co-tenant modules on the FPGA. Finally, we provide a detailed description of the bitstream reconfiguration process and the attack setup on the FPGA.

##### A. Fault Injection Using Power-Wasters

As demonstrated in [8], [12], ROs can be implemented as malicious power-wasters, causing high voltage fluctuations when triggered at a specific toggling frequency. Alternatively, loop-free self-clocked ROs bypass DRC and are capable of generating clock glitches and severe voltage fluctuations on the PDN [16]. Based on these observations, we evaluate both combinational and loop-free ROs for injecting faults into bitstreams. To induce malicious fault injections in bitstreams during partial reconfiguration, it is crucial to determine appropriate fault-injection parameters for the RO grid: (1) Number of ROs, (2) toggling frequency, and (3) duty cycle. We choose the upper bound of the number of RO and self-clocked RO instances to be 16000 and the toggling frequency range as  $10^5 - 10^6$  Hz to be consistent with the experimental setup in prior work [12], [22]. These power-wasters are deployed on the FPGA to launch fault attacks on user bitstreams while they are being loaded into the RM for partial reconfiguration.

##### B. Proposed Fault Attack Methodology

Prior fault attacks [8], [9] require the power-wasters to be activated continuously throughout the runtime of user modules to ensure a successful fault attack. This significantly increases the attack *exposure window*, i.e., the duration for which the attack needs to be active, making it more susceptible to detection. Unlike previous approaches, FLARE is precisely timed to be activated only when the configuration address of the bitstream is being loaded to the RM and remains inactive otherwise. This targeted activation makes the attack stealthy and minimizes the likelihood of detection [13], [14]. By injecting faults in ‘select’ part of the bitstream, FLARE manipulates the configuration address, redirecting the bitstream to incorrect PRR(s), which we refer to as *victim* PRRs.

**Injecting Faults to the Reconfiguration Address:** To understand how FLARE operates, it is important to examine the structure of an FPGA bitstream. Fig. 2 shows the bitstream structure of a Xilinx 7-series FPGA. The bitstream is partitioned into several structured frames [23]. The first section is the synchronization header, which initializes the bitstream. The header is followed by configuration address frames, which allocate a fixed set of bits namely ‘select’ [13]. Finally, the footer includes CRC values. The ‘select’ part identifies the target PRRs where the bitstream is intended to be uploaded. The precise timing of FLARE is based on the fixed and well-defined structure of the Xilinx FPGA bitstream [13]. The configuration process of a bitstream on the FPGA operates at a specific clock frequency. Using this frequency and information about the number of frames prior to the ‘select’ frames, an attacker can estimate the time window during which the ‘select’ part will be configured. In this duration, the attacker activates the RO grid to inject faults in the bitstream.

**Analysis of Attack Impact:** We emphasize that while earlier attacks primarily focused on single-tenant vulnerabilities or localized faults within a specific module, FLARE exposes a broader vulnerability by targeting multiple modules co-located on the FPGA, a threat that was not explored in prior fault attacks [9], [12], [17], [24]. Importantly, we evaluate the effectiveness of the attack after deactivating the power-wasters. At this point, the faulty bitstreams are already loaded into the FPGA, and any errors in module functionality are observed post-reconfiguration. The objectives of FLARE are as follows:

- *Denial-of-service:* By manipulating the configuration address of the user bitstream, the bitstream is redirected to incorrect PRRs. This results in an incomplete configuration of the intended module, rendering it non-functional.
- *Faulty computation in co-tenant module(s):* When a bitstream is redirected to a wrong configuration address, it may overwrite the existing logic of a co-tenant module. This leads to erroneous computations and incorrect functionality of the module. For instance, arithmetic modules used for data processing may generate incorrect results, potentially compromising the integrity and functionality of user modules. Note that once the faults are injected in the victim PRRs, they persist until the FPGA is fully reconfigured.

##### C. Attack Setup

**Dynamic Partial Reconfiguration Using CoRQ:** We utilize an open source runtime RM, namely Command-based Reconfiguration Queue (CoRQ), for loading and dynamically reconfiguring bitstreams on the FPGA [19]. CoRQ can store multiple partial bitstreams corresponding to different modules, and reconfigure them onto specific PRRs of the FPGA based on their configuration addresses. The steps for dynamic reconfiguration of a bitstream are illustrated in Fig. 3. First, the FPGA bitstreams are loaded into the CoRQ memory (marked as *sel\_bitstore*). Next, they are passed through CRC evaluation (marked as *crc\_check*). If CRC passes, the bitstreams are successfully configured on the FPGA, else they are blocked from reconfiguration. Vivado allows users to enable or disable the

```

Input: Input FPGA bitstream  $T$ 
Output:  $corq\_status$ 
/*Indicates if bitstream successfully passes CRC*/
sel_bitstore( $T$ ) /*Load decrypted/unencrypted bitstreams to CoRQ*/
corq_status = crc_check( $T$ ) /*Run CRC test after loading
bitstream*/
if corq_status == 'false' then
| print('Error in loading bitstream')
end
/*Fault attack is detected*/ else
| print('Successfully loaded bitstream')
end
/*No attack detected; bitstream ready for FPGA configuration*/
return corq_status

```

Fig. 3: Pseudocode for partial reconfiguration of bitstreams on the FPGA via CoRQ.

CRC of bitstreams based on their requirements. In scenarios where the bitstream is encrypted, the CRC is disabled using the constraint *BITSTREAM.GENERAL.CRC Disable* [20]. To assess the impact of FLARE on CRC, we deliberately enable CRC during bitstream generation in all our experiments.

CoRQ shares its status as a register that can be read by the processing system, enabling real-time guarantees that can be exploited by attackers. By monitoring when CoRQ starts to be busy, we can precisely determine when the configuration address begins to be uploaded (explained in Section IV.B).

Note that while our experiments target a specific RM, the attack methodology is generic and applicable to various RMs in FPGAs as the reconfiguration always has the same flow.

**Implementation of Victim Modules:** We evaluate FLARE on two design modules: (1) Adders, which are commonly used in ALUs and signal processing, and (2) AES, which is needed in communication and cryptography operations [24]. To create a realistic attack scenario, we configure a majority of the lookup tables (LUTs) on the FPGA with these modules [25]. This reflects real-world multi-tenant FPGAs, where shared modules boost throughput and utilization [26].

### 1) Case Study 1: Adder

In our first case study, we evaluate combinational adders as their outputs are not impacted by timing violations commonly associated with fault attacks [13], allowing us to specifically focus on the impact of address manipulation. To identify which adders are affected by the fault-injected input bitstream, we implement two adder clusters, referred to as adder cluster #1 and adder cluster #2, with each cluster containing  $n$  adder modules. Through experimental analysis, we observe that initializing adders separately rather than grouping them into clusters is not feasible, as it results in inefficient resource utilization. To ensure maximum LUT usage, we set  $n = 500$ ;  $n > 500$  causes placement error during Vivado implementation. Each adder module computes the sum of fixed inputs. In an address manipulation-based fault attack, the input bitstream is redirected to the adder PRRs, leading to erroneous computations in the adder modules. To localize the faulty adders, we generate signals  $flag_1^i$  and  $flag_2^i$ ,  $1 \leq i \leq n$ , corresponding to each adder module  $i$  in adder clusters #1 and #2, respectively. A value of '0' for  $flag_1^i$  ( $flag_2^i$ ) indicates

TABLE II: Resource utilization of different fault attack setups on Pynq-Z1 FPGA (SC: Self-clocked RO).

Module	No. of instances	LUT configured by power-wasters (%)		Overall LUT utilization (%)	
		RO	SC	RO	SC
Adder	1000	29.6	28.2	64.6	76.9
AES	2	22	26	76.7	78.7

that adder  $i$  in adder cluster #1 (adder cluster #2) produces the correct output, while a value of '1' indicates a faulty output.

Monitoring the  $flag_1$  and  $flag_2$  signals directly is not practical due to their large widths of 500 bits each. To address this, we encode  $flag_1$  and  $flag_2$  using priority encoder modules  $p_1$  and  $p_2$ , respectively; this enables us to localize the faulty adder modules within the respective adder clusters. The adder clusters and priority encoders contribute to 21.2% of LUT utilization. Finally, we read the data from  $fltsig$ , which encodes the information about the faulty adders. Specifically, the lower 10 bits of  $fltsig$ , denoted as  $fltsig[9 : 0]$ , are used by  $p_1$  to indicate the faulty adder from adder cluster #1. Similarly, the next 10 bits,  $fltsig[19 : 10]$ , are used by  $p_2$  to denote the faulty adder from adder cluster #2.

### 2) Case Study 2: AES

In our second case study, we evaluate FLARE on an AES-128 implementation. AES encryption and decryption involve the following operations – *sub\_bytes*, *mix\_columns*, *shift\_rows*, and *key\_expansion*. Given the substantial overhead of each AES module in terms of the number of configured LUTs on the FPGA (each AES module utilizes 10223 LUTs), we implement only two such AES instances, namely AES #1 and AES #2, which contribute to 38.6% of the FPGA LUT utilization. For our experiments, we configure both the AES modules with the same 128-bit plaintext and 128-bit key. A fault attack is successful when the fault-injected bitstream is redirected to atleast one of the AES modules, resulting in computational errors. We compare the outputs of the AES modules against the predetermined ciphertext. If the ciphertext of AES #1 (AES #2) matches the expected value, we set  $flag_1$  ( $flag_2$ ) as 0, else we set it as 1. The fault detection signals  $flag_1$  and  $flag_2$  are encoded in the least significant bits  $fltsig[0]$  and  $fltsig[1]$ , respectively. The value of  $fltsig$  determines one of the following:

- $fltsig = 0$ : No fault attack is detected in the AES modules.
- $fltsig = 1$ : A fault attack results in incorrect reconfiguration of a bitstream in the PRR of AES #1.
- $fltsig = 2$ : A fault attack results in redirection of the bitstream to AES #2.
- $fltsig = 3$ : Both AES #1 and AES #2 are impacted.

## V. EXPERIMENTAL RESULTS

### A. Experimental Setup

We implement FLARE on the Pynq-Z1 FPGA, which consists of 53,200 LUTs. The power-wasters, AES, and adder modules are implemented in Verilog. Note that while our evaluation focuses on RO and self-clocked RO circuits, the attack can be demonstrated using other power-wasters discussed in prior literature, as they share similar structural configurations

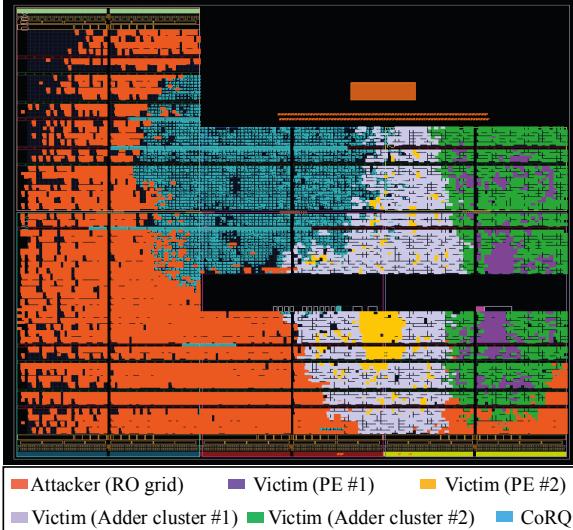


Fig. 4: Floorplan of the attack setup incorporating adder

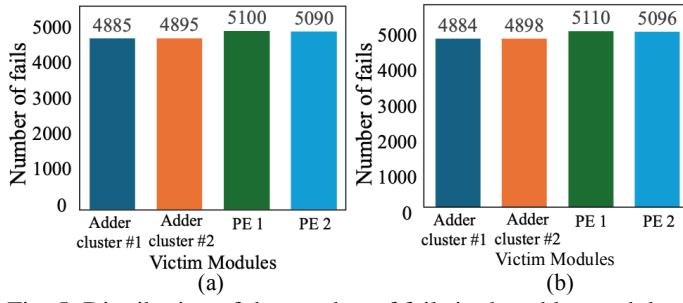


Fig. 5: Distribution of the number of fails in the adder modules for (a) RO and (b) Self-coded RO (PE 1 and PE 2 refer to priority encoders  $p_1$  and  $p_2$ , respectively).

[9] [18] [21]. Importantly, FLARE is generalizable to other FPGAs as the vulnerabilities exploited during the partial reconfiguration process is not FPGA-specific. We use Vivado 2023.2 to synthesize, implement, and generate the bitstream corresponding to the hardware setup for the attack evaluation. Table II shows the resource utilization of the implemented designs. We use Vitis to load partial bitstreams into CoRQ. We test the fault attack using three partial bitstreams – *blinkall*, *blinkcount*, and *blinkline*. We monitor the attack success using the fault detection signal  $flt_{sig}$ , which is captured by the GPIO port of the FPGA. A video demo of the fault attack experiments performed in this work is available in [27].

#### B. Fault Attack Analysis on Adders

We evaluate FLARE on adder modules that are pre-configured on the FPGA. Fig. 4 shows the floorplan of the experimental setup implementing adder modules on the Pynq-Z1 FPGA. We conducted 10,000 fault attack attempts on the FPGA. In each attempt, a different partial bitstream is loaded into CoRQ for configuration. Power-wasters were activated specifically during the reconfiguration process to enable fault injection and were deactivated once the process concluded. After configuring the partial bitstream on the FPGA, we verify that the fault is successfully injected, altering the configuration

address and redirecting the bitstream to the adder PRRs. Fig. 5 demonstrates the distribution of the fails due to the fault attack. We observe that for both combinational and self-coded ROs, FLARE consistently induces faulty computations in at least one of the adder clusters [27]. In some cases, the adders remain unaffected, producing correct outputs. However, the faults disrupt the functionality of modules  $p_1$  and  $p_2$ , resulting in incorrect encoded outputs. Fig. 6(a) and Fig. 6(b), show the frequency of fails in adder clusters #1 and #2, respectively.

#### C. Fault Attack Analysis on AES

To demonstrate the generalizability of FLARE, we evaluate another critical module, namely AES. The attack setup is shown in Fig. 7, which details the placement of the AES instances on the Pynq-Z1 FPGA. We keep the upper bound of AES instances to 2 as adding more instances exceeds the number of LUTs of the FPGA, causing place-and-route errors during the Vivado implementation phase. We performed a total of 10,000 fault attack attempts using both combinational and self-coded ROs. We observe successful fault-injection by both the power-wasters – after five fault-injection attempts, AES #1 consistently generates incorrect ciphertexts, indicating successful disruption of its encryption operations; a demo is presented in [27]. However, AES #2 remains unaffected by the fault attack, primarily due to the placement and routing constraints of the PRRs where it is configured.

#### D. Bypassing CRC Mechanisms

During bitstream generation, the CRC value is computed and embedded within the bitstream. The *crc\_check* function in CoRQ computes the CRC of the bitstream before configuring it on the FPGA. As explained in [28], CRC only verifies the bitstream data; there are no checks to ensure that the bitstream is configured in the correct PRR. To validate whether FLARE evades CRC, we conducted experiments using a test bitstream, namely *blinkcount*. We monitor the *corq\_status* flag to check if CRC detects a faulty bitstream. We obtain the following message on Vitis: *Decrypting blinkcount successful. Bitstream successfully loaded*. Since FLARE manipulates the configuration address rather than the bitstream data, it successfully evades CRC, and the fault-injected bitstream is incorrectly configured in the victim PRR of the FPGA.

#### E. Comparison to Prior Fault Attack Schemes

By precisely targeting the configuration address of a bitstream and reducing the attack duration, FLARE is significantly faster and stealthier than prior runtime fault attacks [8] [9] [10] [11], while minimizing the likelihood of detection by [13] [14]. Moreover, FLARE adversely impacts multiple modules on the FPGA simultaneously, a threat not explored in prior fault attacks. For a fair evaluation, we compare FLARE with [12], as both approaches specifically target fault-injection in the partial reconfiguration process. As highlighted in Table III, FLARE is 200 $\times$  faster than [12] in executing a successful attack. While both [12] and FLARE successfully induce faults in the bitstream, [12] has a 100% detection rate by CRC, whereas FLARE bypasses CRC verification entirely.

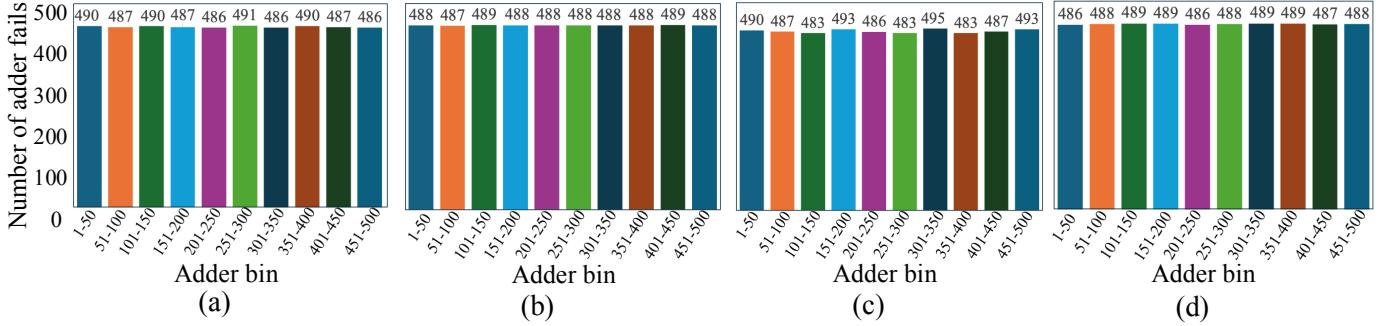


Fig. 6: Parts (a) and (b) depict the number of fails in adder clusters #1 and #2, respectively, with ROs. Parts (c) and (d) show the fail frequency in clusters #1 and #2, respectively, with self-coded ROs.

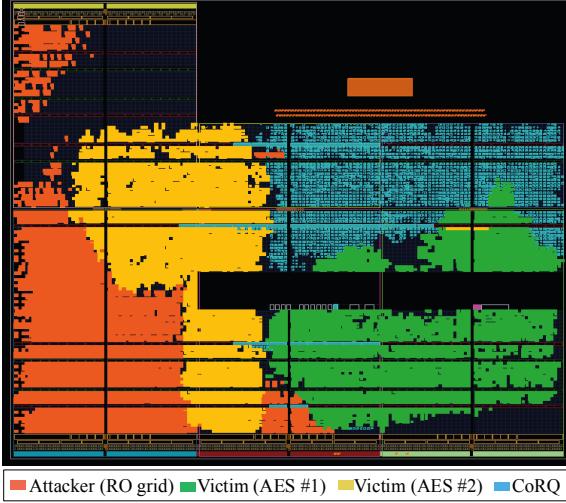


Fig. 7: Floorplan of the attack setup using AES modules.

TABLE III: Comparison of FLARE with [12].

Parameter	[12]	FLARE
Attack duration	40 ms	200 $\mu$ s
Exposure window	Entire bitstream	Targeted ('select' part of bitstream)
CRC evasion?	$\times$	$\checkmark$

## VI. CONCLUSION

We have presented FLARE, a novel fault attack that specifically targets the process of partial reconfiguration of a bitstream, and manipulates the configuration address of the bitstream. Experiments on several design modules demonstrate the effectiveness of FLARE in causing faulty computation by these modules as well as DoS. Simultaneously impacting multiple tenants while reducing the chance of detection highlights the severity of FLARE in multi-tenant FPGA environments.

## ACKNOWLEDGEMENT

The work of J. Chaudhuri and K. Chakrabarty was supported in part by the National Science Foundation under grant no. CNS-2310142. The work of Hassan Nassar was supported in part by the German Federal Ministry of Education and Research (BMBF) through grant 01IS23066 as part of the Software Campus Project “HE-Trust” and in part by the “Helmholtz Pilot Program for Core Informatics (kikit)” at KIT. The work of Mehdi Tahoori was supported by German Research Foundation (DFG) projects SAUBER and SecFShare.

## REFERENCES

- [1] Amazon, “Amazon EC2 F1 Instance,” <https://go.aws/3ENtUj9>, 2021.
- [2] D. Firestone *et al.*, “Azure accelerated networking: SmartNICs in the public cloud,” in *15th USENIX (NSDI)*, 2018, pp. 51–66.
- [3] A. Moradi *et al.*, “On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs,” in *CCS*, 2011, pp. 111–124.
- [4] F.-X. Standaert *et al.*, “Power analysis attacks against FPGA implementations of the des,” in *FPL*. Springer, 2004, pp. 84–94.
- [5] F. X. Standaert *et al.*, “Updates on the security of FPGAs against power analysis attacks,” in *ARC*, 2006.
- [6] R. Elnaggar and K. Chakrabarty, “Machine learning for hardware security: Opportunities and risks,” *Journal of Electronic Testing*, vol. 34, pp. 183–201, 2018.
- [7] S. Trimberger and S. McNeil, “Security of FPGAs in data centers,” *Proc. IEEE IVSW*, pp. 117–122, 2017.
- [8] D. Gnad *et al.*, “Voltage drop-based fault attacks on FPGAs using valid bitstreams,” in *FPL*, 2017, pp. 1–7.
- [9] J. Krautter *et al.*, “FPGAHammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES,” 2018.
- [10] P. Swierczynski *et al.*, “Bitstream fault injections (BiFI)—automated fault attacks against SRAM-based FPGAs,” *IEEE TC*, 2018.
- [11] M. M. Alam *et al.*, “RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions,” in *FDTC*, 2019, pp. 48–55.
- [12] J. Chaudhuri *et al.*, “Hacking the fabric: Targeting partial reconfiguration for fault injection in FPGA fabrics,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.16497>
- [13] H. Nassar *et al.*, “LoopBreaker: Disabling interconnects to mitigate voltage-based attacks in multi-tenant FPGAs,” in *ICCAD*, 2021.
- [14] G. Provelengios *et al.*, “Mitigating voltage attacks in multi-tenant FPGAs,” *ACM TRETS*, vol. 14, no. 2, 2021.
- [15] T. Huffmire *et al.*, “Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems,” in *SP '07*, 2007, pp. 281–295.
- [16] T. Sugawara *et al.*, “Oscillator without a combinatorial loop and its threat to FPGA in data centre,” *Electronics Letters*, 2019.
- [17] G. Provelengios *et al.*, “Power distribution attacks in multitenant FPGAs,” *TVLSI*, vol. 28, no. 12, pp. 2685–2698, 2020.
- [18] J. Krautter *et al.*, “Mitigating Electrical-level Attacks towards Secure Multi-Tenant FPGAs in the Cloud,” *ACM TRETS*, vol. 12, no. 3, 2019.
- [19] M. Damschen *et al.*, “CoRQ: Enabling runtime reconfiguration under WCET guarantees for real-time systems,” *ESL*, 2017.
- [20] Xilinx, “Ultrascale architecture configuration,” <https://bit.ly/2FGfKq4>.
- [21] J. Chaudhuri and K. Chakrabarty, “Detection of malicious FPGA bitstreams using CNN-based learning,” in *ETS*, 2022.
- [22] D. R. Gnad *et al.*, “Remote electrical-level security threats to multi-tenant FPGAs,” *IEEE Design & Test*, vol. 37, no. 2, 2020.
- [23] K. D. Pham *et al.*, “BITMAN: A tool and API for FPGA bitstream manipulations,” in *DATE*, 2017, pp. 894–897.
- [24] J. Krautter *et al.*, “Remote and stealthy fault attacks on virtualized FPGAs,” in *DATE*. IEEE, 2021, pp. 1632–1637.
- [25] Y. Luo *et al.*, “Stealthy-shutdown: Practical remote power attacks in multi-tenant FPGAs,” in *ICCD*, 2020.
- [26] J. M. Mbongue *et al.*, “Architecture support for FPGA multi-tenancy in the cloud,” in *ASAP*, 2020.
- [27] “Fault attack demo,” <https://t.ly/9f3eZ>.
- [28] F4PGA, “Bitstream format,” <https://shorturl.at/N7L2x>.