A Lightweight Embedded Detection System against Voltage Drop Fault Attacks in Multi-Tenant FPGAs

Gwenn Le Gonidec[1][0009−0004−4774−6886], Guillaume Bouffard[2][0000−0002−2046−369X], Jean-Chirstophe Prévotet[3][0000−0001−6951−4702], and Maria Méndez Real[1][0000−0002−9336−9936]

[1] Univ. Bretagne-Sud, Lab-STICC UMR CNRS 6285, Lorient, France
owen.le-gonidec@univ-ubs.fr
maria.mendez-real@univ-ubs.fr
[2] National Cybersecurity Agency of France (ANSSI), Paris, France
guillaume.bouffard@ssi.gouv.fr
[3] UnivRennes, INSA Rennes, CNRS, IETR-UMR 6164, Rennes, France
jean-christophe.prevotet@insa-rennes.fr

Abstract. Voltage drop fault attacks pose a new security threat to FPGAs, especially in a multi-tenant context where several users share the same hardware space. An increasing amount of research shows that attackers can implement voltage plundering circuits to inject timing faults in surrounding hardware modules. In recent years, various countermeasures against such attacks have been proposed. Many of these rely on embedded sensors to detect voltage drops during runtime. These can be effective in disabling the attacker, pinpointing their location or limiting the impact of faults on the victim module. However, the voltage drop detection methods they use are either unsuitable for real-world use as they rely on static thresholds on the sensor's output, or use a significant amount of memory resources. In addition, some lack concrete implementations to prove their effectiveness and to accurately estimate their impact on the targeted system. In this work, we introduce a fast, lightweight and efficient method based on arithmetic calculations for detecting voltage drop attacks. Experiments on a real implementation demonstrate its effectiveness in detecting voltage drops in contexts where methods based on static thresholds would be unsuitable due to noise and uncertainty regarding the sensors' location. Moreover, this new detection method does not require extensive use of memory resources to compute long-term metrics. Its simplicity and versatility makes it easy to integrate into existing countermeasure schemes.

Keywords: Hardware Attacks and Countermeasures · Voltage Drop Attacks · Fault Injection Attacks

1 Introduction

Field-Programmable Gate Arrays (FPGAs) are highly versatile and powerful devices that are particularly well-suited to implement hardware accelerators for

parallel functions. For this reason, cloud service providers such as Amazon Web Services, Tencent and Huawei, have started offering FPGA rental services [27]. As cloud computing often makes use of virtualization to split physical resources between several tenants for utilization optimization purposes, a new research field has emerged, studying the possibility and challenges involved in multi-tenant schemes for cloud FPGAs. Security concerns have quickly become major factors preventing the deployment of the multi-tenancy model. Amongst these concerns are Denial-of-Service (DoS) and fault attacks based on voltage drops. An increasing number of papers describe hardware attacks in which a malicious tenant implements power-wasting circuits that provoke a steep voltage drop. The resulting voltage fluctuations can be used to inject timing faults in the hardware modules of other tenants [1, 10, 16], provoke a crash of the host FPGA [14, 23] or create a covert communication channel [18]. Several works even demonstrate cross-component attacks, in which circuits located in the FPGA are used to carry out a fault attack on the Central Processing Unit (CPU), propagating the voltage drop across the entire Power Delivery Network (PDN) [7, 17].

Many countermeasures against voltage drop attacks have been proposed in the literature. In particular, much work has been done on the detection of voltage drops during runtime. These methods rely on sensors implemented within the FPGA. Most of these countermeasures are based on the assumption that the sensor operates in a static, pre-determined environment [15, 21, 22, 25]. Although they offer a fast detection time and require little hardware resources, this assumption renders them unsuitable for multi-tenant environments, where the location of sensors, surrounding modules and background noise are uncertain. Some other works have attempted to address this issue by proposing methods based on arithmetic calculations that remove the assumption of a static environment [9, 19]. However, these arithmetic methods lack in concrete implementation. They also all rely on storing hundreds of sensor samples to compute averages over long periods of time, which would make them resource-hungry in a real-world implementation. Finally, these methods are designed to localise attackers attempting DoS attacks, which require a prolonged voltage drop [22]. It is unclear whether they can detect short voltage variations intended to cause a timing fault in nearby hardware modules.

In this work, we investigate the detection method against voltage drop attacks in multi-tenant FPGAs. We present a novel detection method that is suited to models where the locations of the attacker and the victim are not be predetermined. It is also lightweight, and fast enough to detect very short voltage drops intended to cause timing faults rather than board crashes. These properties make it compatible with most existing countermeasures. Its lightweight and calibration-agnostic properties make it suitable to localize the attacker by deploying multiple sensors [9, 19]. It is fast enough to be integrated in attacker neutralization schemes [22] and victim protection methods [15]. The main contributions of our work are the following:

1. The limitations of the most commonly used detection methods against voltage drops are demonstrated using experimental data obtained from real-time implementations of attacks.
2. A new detection method based on running variance is proposed. It is implemented in real-time, and is shown to be robust against variations in the sensor location and noisy operating conditions.
3. Short attacks that remain detectable for only a few nanoseconds are implemented. They are shown to be able to introduce timing faults in cryptographic systems. The ability of the variance-based system to detect these so-called short attacks is demonstrated.
4. Real-world data obtained from an FPGA implementation shows that this detection method requires only a moderate amount of resources, comparable to those of the sensor itself. Unlike similar methods, it does not rely on storing hundreds of precedent measurements.

To ensure the reproducibility and availability of our detection system, all of the hardware modules and test procedures used in this work are open-source [13].

The remainder of this article is organized as follows. Section 2 provides an overview of previous work, highlighting the limitations of existing voltage drop detection methods. Section 3 describes our novel detection method in detail and Section 4 introduces the experimental setup used to evaluate our variance-based approach. Section 5 presents the results of these experiments and demonstrates the effectiveness of our detection method against voltage plundering attacks. We also compare it to existing approaches, showing the limitations of the static-threshold-based methods. Finally, Section 6 concludes this work and provides insights for future works.

2 Background and Related Work

In this section, we introduce the relevant prior work on multi-tenant FPGA voltage drop attacks and their detection. We delve into the different types of FPGA voltage sensors and detection methods explored in the existing literature. We emphasise the limitations of existing work, which will serve as a comparison baseline for our contribution.

2.1 Voltage Drop Attacks in FPGAs

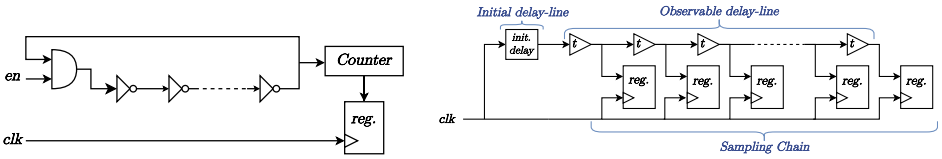
Historically, potent attack methods such as fault injection have required attackers to have physical access to the target system. However, in recent years, several attack schemes that enable remote fault injection have been discovered. Many of them are based on manipulating the voltage and/or frequency of the target component. Amongst them, voltage drop attacks triggered by malicious FPGA circuits have grown in popularity [12]. Most of these attack scenarios consider the fabric to be split between several tenants, two of whom are the attacker and the victim. The attacker uses power-hungry circuits to create a voltage drop around its occupied regions. Their goal is to provoke timing faults in a nearby

module. They can also achieve a voltage-drop-induced crash, or DoS, which involves subjecting the board to more severe and prolonged voltage drops than are required for fault injection [22].

Various primitives can be used as attack circuits. The simplest one is an Ring Oscillator (RO), a simple self-oscillating circuit consisting of `AND` and `NOR` gates [10]. Previous work has showcased attacks based on memory collisions [1], latches, Flip-Flops [26] and cryptographic primitives [24]. When used in large amounts, all of these circuits create strong switching activity that produces transient voltage drops. Consequently, a large part of countermeasures against such attacks embed voltage sensors in the FPGA to detect them.

2.2 Real-Time Voltage Sensors

A key difference in existing approaches is the type of embedded voltage sensor used. Two main types of sensors exist: ROs-based [22, 25], and Time-to-Delay Converter (TDC)-based [9, 15, 21, 22]; as introduced in Figure 1. The former requires few resources, making it suited for scenarios where many sensors are used. The latter, while heavier in resource usage, is known to have better time resolution and less noise in its output value, making it especially suited for transient voltage drop detection [20].



(a) RO-based voltage sensor. (b) TDC-based voltage sensor.

Fig. 1: Sensor types used in voltage drop detection.

RO-based sensors. The frequency of an RO's output oscillation is entirely dependent on the propagation time of its logic gates. Therefore, monitoring this frequency with a counter directly gives a measurement of the current operating voltage. Figure 1a shows the principle of a simple RO-based sensor. It requires few resources, but provides coarse-grained and noisy measurements compared to the TDC-based sensors. It also requires longer sampling periods. For example, in [19], the RO-based sensors achieve a sampling period of 1.25 µs. This is orders of magnitude greater than the time required for an attacker to successfully carry out a power-plundering fault attack, *i.e.* a few nano-seconds [22].

TDC-based sensors, as shows Figure 1b, consist of two chains which are both fed by a clock signal. The first of these chains consists of many primitive elements which one after another propagate an input clock signal. It is divided into an *observable* delay-line, usually made of carry-chains, and an *initial* delay-line, usually made of Look-Up Tables (LUTs). The second chain, referred to as the *sampling chain*, registers the output of each element in the observable delay-line. The initial delay-line is used to introduce a phase shift between the clock used in the sampling chain (the *sampling clock*) and the clock arriving at the first

element of the observable delay-line (the *driving clock*). For the TDC to produce a relevant signal, the initial delay-line has to be calibrated. If it is too small, then the driving clock may propagate through the entire observable line. Conversely, if it is too large, then the driving clock never reaches the observable line. In such cases, the output signal will consist only of `'1'` or `'0'`. Compared to RO-based sensors, TDCs provide short sampling periods (usually one measurement per clock cycle) and have a greater precision. However, their implementation requires more hardware resources. Besides FPGA voltage drop attacks, TDC sensors have been used to detect many different types of attacks targeting FPGAs, such as laser and electromagnetic fault injection attacks [4, 6]. Therefore, their performances and practical usability have been widely investigated.

2.3 Attack Detection Criteria

Existing work explores various uses of these sensors to detect voltage plundering attacks. We classify them into two categories: methods based on static thresholds [15, 21, 22, 25] and methods based on arithmetic calculations [9, 19]. Both aim to sense steep changes in propagation times indicative of an ongoing voltage drop attack. Table 1 summarizes the main strong and weak points of each detection method proposed in existing literature.

Table 1: Comparison of state-of-the-art attack detection methods.

Attack Detection Criteria	Location-Agnostic	Online Detection	Single Sensor Needed	Complexity	Short Attack Detection
Static Threshold [15, 22, 25]	✗	✓	✓	–	✓
Variation from the average value [19]	✓	✗	?	+ + +	?
Accumulated Hamming Distance [9]	✓	✓	?	++	✗
Objectives of this work	✓	✓	✓	+	✓

Static threshold method. Approaches based on static thresholds simply use fixed limits at the output of the sensor, which detect an attack whenever they are exceeded. This method is straightforward to implement. When used with a TDC sensor, this method can detect attacks with a minimal delay of just a few nanoseconds [22]. However, it has important limitations. The most prevalent one is that the raw output value of the sensor can be affected by several factors, making it unreliable in certain contexts. These factors include the board's manufacturing process, the location of the sensor within the fabric [19], and the overall activity on the FPGA [23], which are difficult to predict in a multi-tenant context. This method is henceforth very vulnerable against Process, Voltage and Temperature (PVT) variations, which can lead to the detection of false positives. More importantly, it assumes that the threshold is determined for a pre-defined and unchangeable position on the FPGA, and for a known activity level in neighboring locations. These limitations are more thoroughly studied and demonstrated in Section 5.2.

Arithmetic methods. Other literature endeavors make use of arithmetic calculations to transform the plain output of the sensor into a metric that can be exploited to detect attacks. In existing approaches, the main objective is to infer the attacker's localization by comparing the output of several sensors at various positions in the fabric. In [19], RO-based sensors are used. The average of each sensor's output is computed over a long window of 500 µs. Its worst-case output on a shorter window of 10 µs is also registered. These two values are compared to estimate the amount of variation of a sensor. A similar approach is described in [9]. It uses TDC sensors, and relies on computing the accumulated Hamming weight at the output of the sensor. Although the sensors themselves are implemented on the FPGA, the authors propose computing the accumulated Hamming distance offline. This makes this method difficult to evaluate for real-time detection.

Although both of these arithmetic approaches suggest promising countermeasure strategies, they may not be suitable for attack detection on a multi-tenant FPGA threat model. First, due to characteristics of the PDN, high switching activity creates voltage fluctuations via two current biases. One is by resistance (IR *drop*) and one is by impedance (di/dt drop). The IR drop produces effects that last for several microseconds, but the di/dt part remains observable for a matter of nanoseconds [5]. Therefore, a sampling period on the order of microseconds, as described in [19], is not sufficient to detect all types of voltage drops. In Section 5.1, we demonstrate that even nanosecond-scale voltage drops can reliably cause timing faults in an Advanced Encryption Standard (AES) hardware module. Another important caveat is that both detection methods rely on storing a significant amount of measurements to compute long-term averages or accumulations. The approach described in [19] involves storing the last 500 µs of data, which represents 400 samples with a sampling period of 1.25 µs. In [9], the accumulated Hamming weight is computed on the last 1024 sensor samples. The large amount of retained data should make these methods resource-intensive. However, information on their resource utilization is not provided.

Arithmetic approaches have also been used to counter other types of voltage-related attacks. In [4], the average of the sensor readings on the last 8 clock cycles is compared to its current cycle value to detect laser fault injection. The authors demonstrate the efficiency of this method in detecting localized and subtle variations in voltage over periods of a few dozen clock cycles, which are typical of laser fault attacks. However, it does not fit the behaviour of the voltage during voltage drop attacks, as described in Section 2.1, which produce steep changes in the voltage observed by the TDC. Additionally, we found that non-attacker hardware modules can provoke a significant background noise in the voltage measurements reported by the TDC. This background noise would result in false positives being detected by a system designed to sense subtle voltage changes. In the context of a voltage drop induced by attacker circuits, there is a need for a method that amplifies steep changes while negating the effect of noise induced by surrounding benign modules.

3 Proposed Lightweight and Dynamic Detection Method

The previous section outlines the limitations of existing detection methods. This section describes our novel approach, which combines the advantages of static threshold methods and arithmetic calculation methods. Specifically, our approach does not assume a fixed baseline value (*i.e.*, sensor output without attack), which makes it location-agnostic and better suited to multi-tenant models than static threshold methods. Additionally, it is based on a metric computed at each clock cycle, enabling it to detect attacks lasting mere nanoseconds. Finally, it does not rely on storing a large number of previous measurements. To achieve this, we use a well-known and simple to implement metric: the variance of the last T sensor samples, which we refer to as running variance.

3.1 Threat Model

To evaluate our countermeasure, we use the following threat model, summarized in Figure 2. We consider an FPGA with several tenants, including two adjacent areas: a malicious tenant controlled by the attacker, and a victim tenant where the detection is implemented. The victim tenant does not fully control its location on the FPGA. Therefore, the exact location of the sensor on the fabric is unknown. The attacker does not have physical access to the target device. They aim to create timing faults in one of the modules of the victim tenants. To achieve this goal, they use attack circuits e.g., based on ROs or other power-hungry circuitry. We assume there is no offline detection for potential attack circuits which could prevent their deployment. This is a realistic hypothesis given that new works regularly demonstrate new circuits that can be used for voltage drop. Countermeasures that rely on preventing the deployment of attack circuits must be constantly updated to detect increasingly sophisticated attack circuits [2]. Without loss of generality, we use hardware AES encryption circuits to materialize the victim. The goal of the attacker may be to compromise the integrity or confidentiality, for instance through Differential Fault Analysis (DFA) [10].

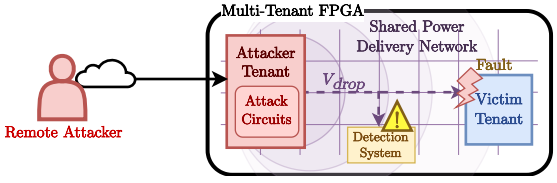


Fig. 2: Considered threat model.

3.2 Detection Method

We consider a sensor that at each clock cycle t outputs an estimation of the propagation time (*i.e.*, supply voltage) $v(t)$. This value depends on a constant baseline level $\overline{v}$ (*i.e.*, its average value without noise and attack), and a noise $n(t)$

due to benign module activity and temperature conditions. We model the variation of voltage created by an attack with a gate function $\Pi(t)$. The propagation time can then be modelled by the formula given in 1.

$$v(t) = \overline{v} + n(t) + \Pi(t) \tag{1}$$

Our metric for attack detection is a running variance on the sensor output. At the current cycle t_0, it gives the amount of voltage variation that occurred during the last T clock cycles. It can be formulated as the difference between each voltage value v_t registered during the last T clock cycles and their average value $v_T = \frac{1}{T}\sum_t v_t$ over the same period of time, as shows Equation 2.

$$\mathbb{V}_T\left(v, t_0\right) = \sum_{t=t_0-T}^{t_0} \left(v(t) - \overline{v_T(t_0)}\right)^2 \tag{2}$$

By replacing the v terms with our propagation model given in Equation 1, we remove the average value $\overline{v}$ from the computation. Indeed, if $\overline{v_T} \approx \overline{v}$ at the current cycle t_0 (*i.e.* when the average value during the last T cycles is close to the constant baseline), then the variance is only affected by noise and attacks, as shows Equation 3.

$$\mathbb{V}_T(v, t_0) = \sum_{t=t_0-T}^{t_0} \left(\overline{v} + n(t) + \Pi(t) - \overline{v_T(t_0)}\right)^2 = \sum_{t=t_0-T}^{t_0} \left(n(t) + \Pi(t)\right)^2 \tag{3}$$

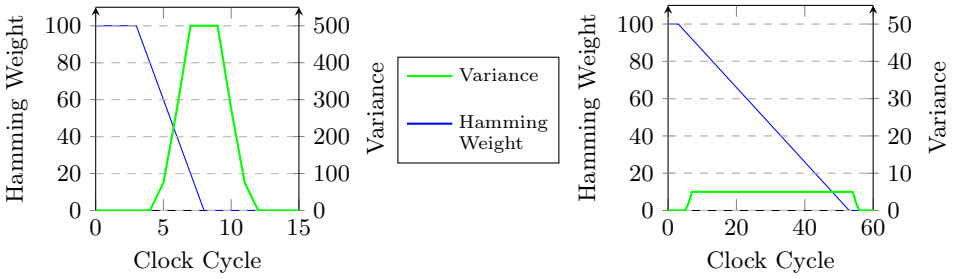
In practice, we use the formula given in Equation 4, which is equivalent to the previous one, to compute the running variance. It is based on the average voltage value $\mathbb{E}_T(v)$ over the time period T, and can be obtained by expanding the squared term in Equation 2. Implementing this computation on an FPGA is fairly simple and only uses additions, multiplications and bit-shifts as long as T is a power of two.

$$\mathbb{V}_T(v, t_0) = \mathbb{E}_T(v^2) - \mathbb{E}_T(v)^2 \approx \frac{1}{T}\sum_{t=t_0-T}^{t} v^2(t) - \frac{1}{T^2}\left(\sum_{t=t_0-T}^{t} v(t)\right)^2 \tag{4}$$

We also tried using Welford's algorithm for the variance calculation method. This is a widely used method for computing an on-the-fly approximation of the variance [28]. We found that it requires around half the hardware resources, but is less responsive to short voltage drops.

We define a detection threshold M as the maximum variance under normal operating conditions, *i.e.* when no attack is occurring. With M high enough for the noise to be negligible, this method can detect sudden voltage fluctuations that could cause a fault without fault positives.

The only controllable parameter in this implementation is the window size T. This parameter is used to cancel the effect of noise and amplify steep variations. However, the higher this parameter is, the more registers need to be implemented to store previous states of $v(t)$. This is the main trade-off of this variance detection system. As described above, variance is efficient at detecting attacks only if $\overline{v_T} \approx \overline{v}$, *i.e.* if the average value during the last T cycles is approximately equal to the baseline value, the value without an attack. As Figure 3a shows,



(a) Steep voltage change.

(b) Slow voltage change.

Fig. 3: Simulated running variance evolution.

this condition is verified at the very beginning of an attack. In other words, variance only detects steep changes in the voltage. However, it is theoretically possible to craft an attack that slowly drops the voltage down to a faulting point. If done stealthily enough, the average term $\overline{v_T(t)}$ rises slowly enough for $\sum_{t_0-T}^{t_0}(v(t) - \overline{v_T})^2$ to be near zero. Figure 3b shows that a slow-rising attack will not produce a significant result on the variance.

Nevertheless, we argue that such a slow-rising attack is hardly practical for the attacker targeting a fault injection. Indeed, the exact timing of a fault provoked by such an attack is too unpredictable for its use in well-known attack schemes such as DFA. To perform such an attack on AES for example, it is necessary to target a specific encryption round (*i.e.* a single clock cycle), which is a level of precision a slow attack would struggle to achieve [10]. These slow attacks can be countered by combining the variance-based system with a simple voltage comparator that has a lower sampling period. This possibility is not investigated in this article.

4 Experimental Setup

This section describes the methodology used to estimate the efficiency of our variance-based detection scheme in real-time. To ensure its efficiency against attacks representative of real-world scenarios, we propose implementing a fault injection attack against a hardware AES module. This attack scheme is frequently showcased in the literature [10, 15]. It will subsequently serve as a reference to evaluate the success rate and latency of our detection method. All experiments described henceforth are realized on the FPGA of a Zynq UltraScale+ ZCU104 board. We use the Vivado tools suite to design, implement and debug all circuits.

4.1 Victim Module

To test the ability of the reference attacks to provoke faults, without loss of generality, we use an open-source AES implementation [8]. It is not pipelined and computes one AES round per clock cycle, taking 10 cycles for a full encryption.

The maximum operating frequency of this module is determined by implementing it on our board's FPGA using different clock frequency constraints, with steps of 25 MHz. For each frequency, we launch the module with known input plaintext and cipher keys and compare the output ciphertext to the expected result. A fault is considered to occur when the two values are not equal. Using this method, we observe that timing faults systematically occur if the module is run above 625 MHz, and don't occur under 600 MHz. For the following tests, we use a frequency of 550 MHz. This clock frequency is low enough for timing faults never to occur within normal operating conditions, even with some non-intentional voltage fluctuations.

4.2 Attack Circuits

In this work, we use state-of-the art RO-based attack circuits comprising several blocks of Enhanced Ring Oscillators (EROs), a type of ROs which produces voltage drops at a low cost in resources [11, 17]. It is considered to be the strongest power waster and has been used to evaluate similar detection methods [22]. We do not investigate other attack circuit types described in Section 2.1 as they are all able to produce similar results from a voltage sensor standpoint, *i.e.* ample voltage drops. The attack circuits we use are similar to those used in recent ERO-based attacks [17]. As Figure 4 shows, individual EROs are grouped by n_{EROs} to form *ERO Blocks*, which are themselves grouped in n_{blocks} to form an *ERO node*. One ERO set comprises n_{nodes} ERO nodes. In all experiments henceforth, we fix $n_{eros} = 10$ and $n_{blocks} = 20$ while n_{nodes} changes for characterization purposes. This structure enables the attack signal to propagate cleanly and ensures that attack circuits are evenly placed.

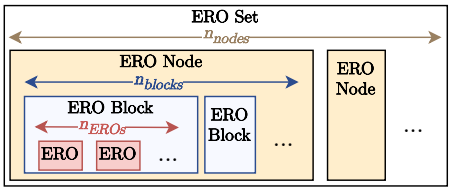


Fig. 4: Structure of the attack circuits.

In previous work, victim circuits are typically considered to run at a higher frequency than the TDC- or RO-based countermeasure [15]. Therefore, we assume that the attacker may also use high-frequency control circuits to produce very short spikes in voltage. In all the following experiments, the attack circuit uses the same clock frequency as the victim module, *i.e.* 550 MHz.

4.3 Detection Modules

To sense voltage drops generated by these attack circuits, we use a TDC sensor alongside the variance-based detection metric described in Section 3.

TDC implementation. The main parameter of a TDC implementation is the size of its observable chain. Previous work has used TDCs of varying sizes,

namely 32 [9], 64 [25] and 512 bits [15]. A large observable chain allows for the measurement of larger voltage variations, but linearly increases the number of resources required. We use an intermediate length of 128 bits for the observable chain in all subsequent experiments. Our board features 8-bit carry-chain (`CARRY8`) primitives. This makes our sensor more fine-grained than those in some previous works that used 4-bit (`CARRY4`) primitives [15] as the input clock signal traverses 8 bits at a time between each net.

Hamming Weight computation. The raw output of the TDC is a 128-bit value. This size makes it difficult to observe in real time. It can be simplified by computing the Hamming weight of the TDC's observation chain, as suggested in previous work [9]. The measurements then use a compact and simple representation that is better suited to calculating variance. Moreover, reduces sample data from 128 bits to 8 bits, thereby making the overall calculations less costly in terms of hardware resources. Thus, in all subsequent experiments, we use the Hamming weight to represent the output of the TDC. It is simply implemented by adding the value of all 128 bits of the sampling chain.

Time Window. As discussed in Section 3, the time window T is an important parameter in our variance calculation method. To simplify divisions into bit-shifts, it must be a power of two. Since higher values imply greater resource usage, it should be as small as possible. We found that using $T = 4$ provides efficient detection, while using a value of 2 is less effective at distinguishing short attacks from noise. Therefore, in all subsequent experiments, we will use a time window of four samples.

Operating frequency. In our implementation, the entire detection system operates at a frequency that differs from that of the attacker and victim modules. Although TDC sensors are known to be suitable for high-frequency measurement [20], we found during our experiments that using ours above 200 MHz produced increasingly noisy and unreliable measurements. Furthermore, the computation time of the variance is the main bottleneck of our detection system, taking multiple cycles to execute. This aspect is discussed in depth in Section 5.5. Thus, increasing this clock frequency is unlikely to significantly reduce the detection delay as the variance will take about the same amount of time to compute, but would increase the uncertainty of the measurement. Thus, in all upcoming experiments, the detection system (TDC sensor, Hamming weight and variance calculation) runs at 200 MHz while the victim AES module and the attack circuit run at 550 MHz.

4.4 Evaluation Methodology

As explained in Section 2.3, detection schemes based on arithmetic calculations use several sensors placed at various locations in the FPGA fabric. Thus, to evaluate the location-agnostic nature of our detection method, we propose to use three identical sensors at different locations, along with the associated detection modules. Following the setup of similar work [15], we divide the attack circuit in two portions, each containing half of the ERO nodes. The circuits are synchronized by an attack control unit, which simply sends the enabling signal to

a specified number of ERO nodes of both sets for a specified duration. Figure 5 summarizes all of the hardware modules we use in our experimental setup.

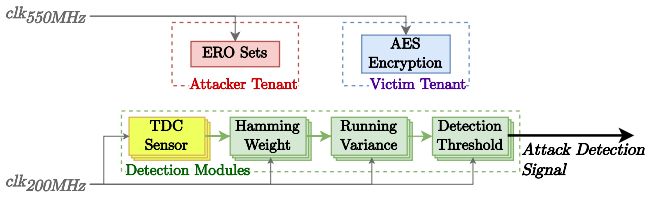


Fig. 5: Modules used in the experimental setup.

4.5 Placement

Figure 6 shows the placement of all aforementioned modules on the FPGA fabric. Using Xilinx development tools, we fixed the placement of the three sensors one by one so that they were implemented in exactly the same way. During our experiments, we discovered that even slight differences in their implementation can result in completely different observed values. Sensors 0 and 1 are placed in opposite corners of the same clock region, which constitutes the optimal placement for attacker detection according to previous work [9], while Sensor 2 is placed in a distant clock region. The two halves of the attack circuits are placed around the sensor and the victim. We designed the attack circuit so that blocks of an ERO node would not be placed next to one another, but rather uniformly throughout the global ERO set. To extract measured data and manipulate our hardware modules, we use debug hardware appliances provided by the Vivado design suite, namely the Integrated Logic Analyzer (ILA) and Virtual Input-Output (VIO) modules.

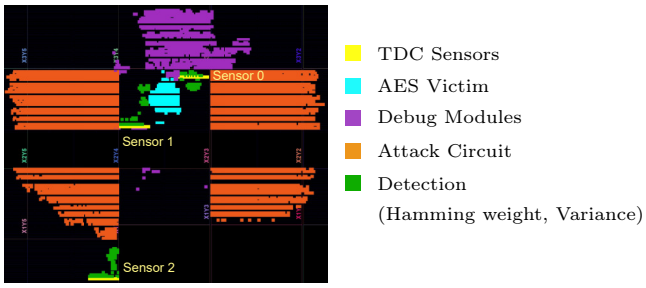


Fig. 6: Placement of hardware modules in the experimental setup.
(Rotated 90 degrees counterclockwise)

5 Experimental Results

In this section, we assess the efficiency of the proposed detection method against voltage drop attacks in a real-time implementation, and compare these results to similar work.

5.1 Attack Characterization

As discussed in Section 2.3, one of the main caveats of existing detection methods based on arithmetic calculations is that they focus on long voltage drops intended to cause DoS attacks. However, they have not been evaluated against nanosecond-scale attacks that aim to provoke timing faults [9, 19], and might not be efficient in this scenario. To ensure our detection method can reliably detect the most transient attacks, we investigate the smallest possible attack circuit size and duration that can cause faults in the victim circuit. To do so, we run attack attempts while changing the number of activated ERO Nodes and the number of clock cycles during which the attack circuit is active. As in previous work, we use the percentage of LUTs used by the attack circuit as our metric for characterising its size. We find that each node occupies 0.4 % of the FPGA's total LUTs. We run one thousand attacks attempt for each amount of LUTs between 12 % and 4 %. An attack attempt consists in activating the ERO circuits for one to five clock cycles. A waiting period of one million clock cycles is observed between each attempt to ensure that the attempts are independent of each other. We define an attack as successful if it produces at least one fault during the AES encryption. The number of successful attacks is measured and compared to the total number of attempts. The results of this experiment are reported in Figure 7. We were not able to observe any successful fault for circuits that occupy 2 % of the LUTs or less. These results show that even very short attacks with small attack circuit size can provoke faults. However, the success rate of some combinations is very low. In particular, single-cycle attacks show poor results below 8 % LUTs. We observed during these experiments that the attack duration is much more impactful on the fault probability than the attacker size, but it also provokes a bigger variation on the TDC output value.

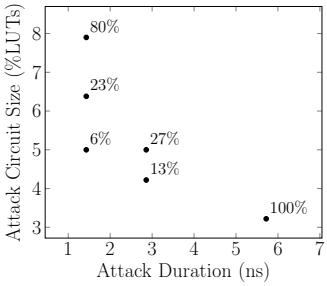


Fig. 7: Fault probabilities with different attack parameters.

Considering these results, it appears that an attack that uses 8 % of the LUTs (*i.e.* 20 active ERO nodes) and lasts for 1 clock cycle at 550 MHz (*i.e.* 1.82 ns) is a good reference point. It reliably faults the victim AES module, with 80 % measured success rate. We note that this attack is much faster than an AES encryption cycle, which is 14.3 ns (*i.e.* 10 clock cycles), making it very suitable for DFA. In the following sections of this article, we designate this configuration as the reference *short attack*. In addition to this short attack, we also define a *normal attack*, closer to what is used in similar work [15]. It uses 34 active nodes,

i.e. 13.6 % of the total LUTs of the FPGA, and is active for 100 clock cycles. Using both the short and normal attacks to evaluate our countermeasure ensures that it can detect various voltage drop attack patterns. Table 2 summarizes the parameters used for our reference attacks.

Table 2: Summary of the parameters used for the reference attacks.

Reference attack	Attack circuit frequency	Active duration	% of LUTs used
Normal attack	550 MHz	100 clock cycles (182 ns)	14 % (34 ERO Nodes)
Short attack	550 MHz	1 clock cycle (1.82 ns)	8 % (20 ERO nodes)

5.2 Limitations of the Static Threshold Method

Before delving into the results obtained with the variance metric, we first consider the direct readings of the sensor represented by the output of the Hamming weight modules. As stated in Section 2.3, most countermeasures simply use a static threshold on the output of the sensor as their main detection criterion. However, our experiments show that this direct reading is not suitable in multi-tenant contexts, as the location of the sensor on the fabric is not fully controllable. Figure 8 shows the baseline value of each of the three sensors, as well as their maximum value under the reference normal attack. Each sensor's baseline value is far apart from the others, and the attack produces roughly the same outline on the three of them. The intensity of the variation displayed by the three sensors is not the same due to the non-linear properties of the TDC, and because the attack can cause them to reach the minimum (`0`) or maximum (`127`) observable values.

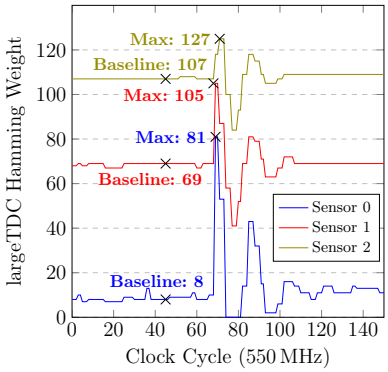


Fig. 8: Output of the three TDC sensors during the reference normal attack.

These results demonstrate that the static threshold method is effective only if the baseline value of the TDC is known and fixed at a pre-determined location. For instance, Figure 8 shows that Sensor 0's under-attack value is close to Sensor 1's baseline value. Therefore, if the threshold configured for the former is used at the latter's location, false positives will be detected. Conversely, if

Sensor 1's threshold is used at Sensor 0's location, it may not detect attacks as the under-attack value will not exceed the baseline as much as it should. We note that both sensors are placed in the same clock region, as Figure 6. These results emphasise the importance of using a location-agnostic metric for detecting voltage drop attacks in a multi-tenant context where the relative position of the attacker tenant and the sensor may not be pre-determined.

5.3 Noise Characterization

Most of the dynamic countermeasures against voltage drop attacks in FPGAs are based on deactivating or removing attack circuits [9, 19, 22]. Therefore, detecting false positives can lead to negative impacts on benign modules. We propose to investigate voltage variations provoked in a scenario where the victim tenant generates a significant amount of activity. This information will be useful to determine a relevant value for our detection threshold M that has a low chance of provoking the detection of false positives. To the best of our knowledge, no similar work (*i.e.* methods presented in Section 2.3) has studied this possibility.

The setup used in this experiment is shown in Figure 9. Two AES modules are placed respectively at the top and at the bottom of the sensor's observable carry-chain, *i.e.* at the closest possible position to the TDC sensor. The experiment consists in simultaneously activating and deactivating them using a `reset` input. When activated, the AES modules use the generated ciphertext as their input plaintext to simulate random input values, thereby ensuring maximum commutation.

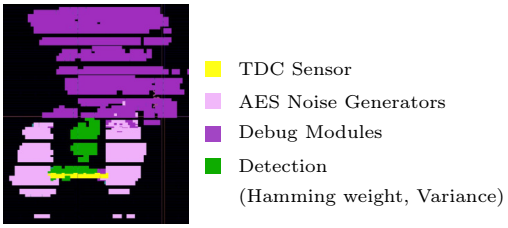
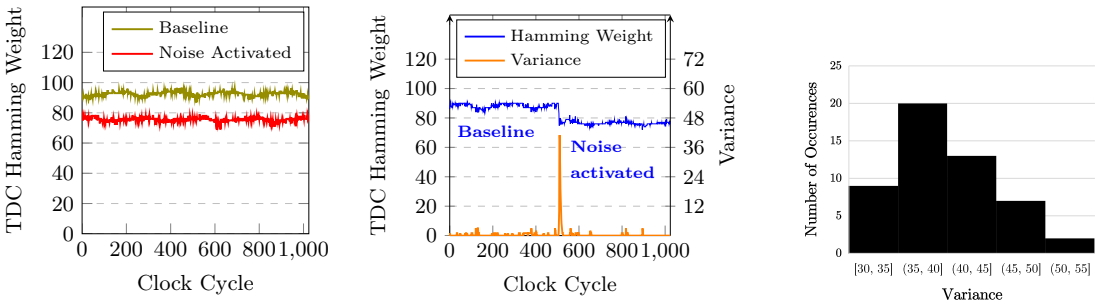


Fig. 9: Placement of hardware modules in the noise characterization experiment. (Rotated 90 degrees counterclockwise)

Figure 10 summarizes the results of this experiment. The baseline value shifts by a constant value when noise is active, as shown in Figure 10a. This is likely caused by resistive components of the PDN, with a higher current load causing a higher average voltage level. In addition to this resistive shift, a low-frequency noise can be seen whether or not the AES modules are activated. This is likely a Simultaneous Switching Noise (SSN) that's due to imperfections in the PDN's bypass capacitors. We empirically found the oscillation frequency of this noise to be around 800 kHz in our experiments. Its value is mainly related to the resonant frequency of the PDN [3]. Thus, noise caused by activity near the sensor can mostly be separated in two components: a constant resistive shift, and a low-frequency sine-shaped voltage ripple. Further experiments were carried



(a) Sensor output with and without active noise modules.

(b) Variation on the sensor upon activating the noise modules.

(c) Maximum variance upon activating the noise modules.

Fig. 10: Results of the noise characterization experiment.

increasing the number of AES modules used to generate noise, but this didn't lead to an observable significant increase in the resistive shift or the SSN, likely because the additional AES modules could not be placed in the same clock region as the sensor due to resource constraints. Thus, it appears difficult to produce more noise than that shown in Figure 10. As shows Figure 10b, the resistive shift occurs immediately upon activation of the noise modules. It produces a steep voltage variation and significantly affects the variance. This resistive shift is the only aspect of the noise that significantly affects the variance. We investigated its effect by collecting data from 50 activations of the noise modules and observing the maximum value of the variance during this activation. Figure 10c shows the results of this experiment. Out of 50 trials, we observed only 3 occurrences of the variance's output going above `50`, the highest value being `55`.

5.4 Threshold Determination

As in previous work, the detection threshold is based on the observation of typical values without and under attack [25]. We note that, unlike static threshold-based methods, this characterisation only needs to be performed once. While we only consider Sensor 1 for this characterization, our detection threshold M should remain valid for other locations on the FPGA. We verify this hypothesis in Section 5.6.

Figure 11 shows measurements done during the reference normal and short attacks. As expected from the theoretical considerations described in Section 3, the variance only increases at the beginning of the attack. During an attack, the variance typically rises above `500`, which is about ten times its maximum value in a noisy environment without an attack, as discussed in Section 5.3.

However, as can be seen in Figure 11b, it takes several cycles for the variance to reach its maximum value. To investigate this phenomenon further, we run 100 short attacks and observe the first value taken by the variance after the attack starts. The results of these measurements are reported in Figure 12a. These tests

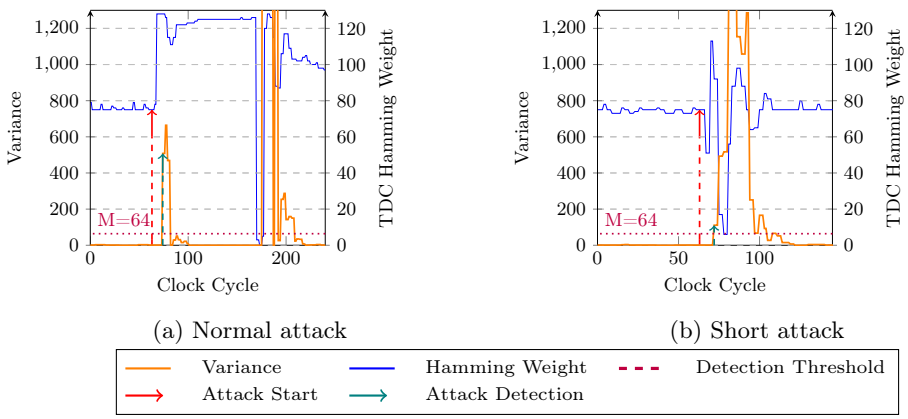
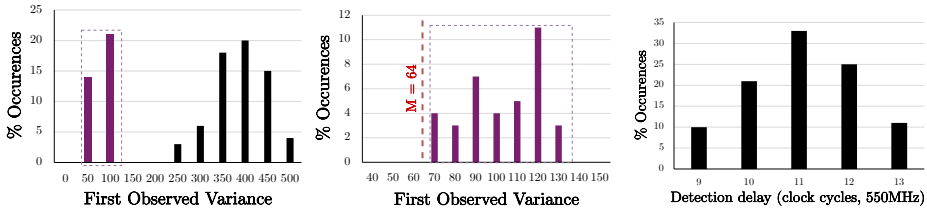


Fig. 11: Detection of the reference short and normal attacks (Sensor 1).



(a) First value of the variance after attack starts. (b) Zoom-in of Fig a) [40-150] (c) Time before $M = 64$ is exceeded.

Fig. 12: Attack detection delay and first value of the variance after the threshold $M = 64$ is exceeded.

show that, after the attack starts, the first value that the variance takes can be sorted into two groups. We found that this is highly correlated with detection delay: attacks detected sooner tend to have a smaller effect on the variance at the clock cycle at which they are detected. In our tests, this situation occurred approximately once in every three measurements (35 % of occurrences). Figure 11b shows a concrete example of this behaviour. Investigating the lowest value the variance takes in such cases, Figure 12b zooms on the group of measurements with the lowest variance. The distribution of the values does not appear to follow a standard probability law. The lowest observed value in these tests is 73.

In summary, we found that activity generated by a legitimate circuit can increase the variance to 55, whereas the initial variance value after an attack begins is 73. Therefore, we propose using the intermediate value $M = 64$. This value should detect all attacks in the shortest possible time without producing false positives. We emphasise that the scenario considered in Section 5.3 is an extreme case, as it involves enabling and disabling entire hardware modules as close as possible to the sensor. Therefore, it is unlikely that a real-world use case will generate as much, or even more, variation in the voltage. Thus, it is unlikely that false positives will be detected.

5.5 Detection Delay and Success Rate

Figure 12c shows the delay from the start of the attack to the moment the variance exceeds M. The data used consists of the 100 attack samples that were used in Section 5.4. Results show that the observed delay ranges from 9 to 13 clock cycles at 550 MHz. During attack characterization, we found that a timing fault typically occurs in the victim AES module in the 1 to 5 cycles following the start of the attack. Therefore, it is unlikely that the variance-based method is able to detect an attack before a fault occurs. However, amongst the other detection methods based on arithmetic calculations, [19] has a significantly slower detection time of 1.25 µs. The other approach based on accumulated Hamming distance is based on off-line calculations, which haven't been evaluated in real-time [9]. Furthermore, previous work shows that there is currently no method of attacker deactivation or victim protection that can be activated before a fault occurs [9, 22]. Consequently, most countermeasures either rely on rollback mechanisms for the victim modules [15] or focus solely on voltage drop attacks that cause the entire board to crash. Indeed, while a fault takes about 1 ns to occur after the attack starts, a crash requires the attack circuit to be active for more than 1 µs with the same attack circuit size [22].

To compute the detection rate of the variance countermeasure, we run 32k short attacks and the same amount of normal attacks, with a waiting delay of one million clock cycles between each attack to ensure no residual state is left from an attack to the next one. We found that our detection method, at each sensor, successfully detected all attacks with the threshold value of $M = 64$ defined experimentally. As shown in Figure 11 for Sensor 1, the variance typically increases to more than `50` during an attack, demonstrating that the variance metric is highly effective in highlighting abnormal behaviours, and is a very reliable indicator overall.

5.6 Impact of the Sensor Location on the Running Variance

In Section 5.2, we demonstrated that static thresholds at the sensors' output are not suitable for use cases involving various sensors at different locations within the FPGA. To demonstrate the efficiency of our approach in such contexts, we study how the running variance behaves under the same conditions. Figure 13 shows the raw output of the hamming weight and the evolution of the variance under the reference short attack, from the same attack attempt as the one shown in Figure 8. Despite the Hamming weight baseline values being very different from each other, and being at the edge of the observable window, the variance goes far above M if and only if an attack is ongoing. These results confirm that our approach removes the need for on-site determination for the detection threshold, in cases where many sensors are deployed on a board. It also shows that variance can detect attacks reliably even if the TDC sensor is poorly calibrated, as long as the signal is within the observable window. Finally, it proves that the value of M, which was determined for Sensor 1, is also valid for other sensor locations.

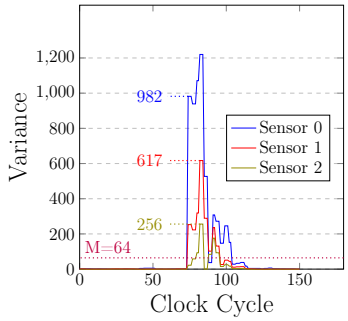


Fig. 13: Variance of the three sensors during the reference short attack.

5.7 Resource Utilization

Unfortunately, the resource utilization of similar detection schemes based on arithmetic calculations is not known [9, 19]. Therefore, a fully quantitative comparison is not possible. However, the aforementioned approaches rely on memorizing hundreds of precedent sensor measurements to compute long-term metrics. Meanwhile, the calculation of the running variance only uses the last four measurements. Furthermore, simplifying the TDC outputs using the Hamming weight enables the running variance to be computed using 8-bit values. Table 3 provides details on resource usage for each module of this countermeasure. The variance calculation logic is the largest contributor, requiring a large portion of the total LUTs used by the detection module overall, and one Digital Signal Processing (DSP) block. However, the sensor itself uses half of the total carry-chain elements and most of the registers. Therefore, compared to using a single sensor with a static threshold, the variance-based method is in the same order of magnitude in terms of resource usage.

Table 3: Resources needed for a single sensor using the variance countermeasure.

Module	**LUTs (%)**	**CARRY8** (%)	**Registers** (%)	**DSP** (%)
TDC sensor	32 (0.01)	16 (0.06)	128 (0.02)	0
Hamming weight	148 (0.06)	1 (< 0.01)	8 (< 0.01)	0
Variance	289 (0.13)	34 (0.12)	48 (0.01)	1 (0.06)
Adder (M)	8 (0.01)	1 (< 0.01)	1 (< 0.01)	0
Total	**477 (0.18%)**	**52 (0.18%)**	**185 (0.03%)**	**1 (0.06 %)**

As stated in Section 4.3, this implementation of our detection method uses a fairly large TDC, with 128 observable bits, compared to some of the previous work which has used 32 [9] or 64 [25]. Since the calculation logic scales with the size of the TDC, using a lighter sensor would linearly decrease the overall resource usage for each module. This decision was mostly driven by the location-agnostic requirement of our detection scheme. Indeed, as shows Section 5.2, using a smaller TDC could lead to signals becoming non-observable, with the output of the TDC being stuck at the minimal or maximal value.

5.8 Results Summary

As stated in Section 3, we aimed to design a detection method that meets the specific needs of FPGA multi-tenancy, able to detect nanosecond-scale fault attack attempts using a relatively lightweight implementation with regard to sensor size. In the previous section, we proved using experimental data that the variance-based scheme is compliant with these requirements. Table 4 shows a comparison of our implementation with previous work. As discussed in Sections 5.5 and 5.7, methods based on static thresholds outperform in terms of detection delay and resource utilization, with single-cycle detection and only a few LUTs used outside of the sensor itself. However, we showed in Section 5.2 that this approach is not suitable for multi-tenant schemes where the location of the sensor is not known and multiple sensors may be required for a single tenant. Detection schemes based on arithmetic calculations are comparable to ours in terms of use case, as they also propose location-agnostic methods. Unfortunately, the lack of information on resource usage and detection time of these approaches does not allow for a fully quantitative comparison. As demonstrated in Section 5.5, our implementation provides a worst-case detection time of 23.6 ns, which is about the same duration of one AES encryption cycle. In addition, we proved in Section 5.1 that the shortest possible attacks, which are enabled for a single clock cycle, are successfully detected by our method. This is mostly due to the sampling period of our method being the shortest achievable, *i.e* one clock cycle. Regarding resource utilization, Section 5.7 shows that hardware resources required for the variance calculation are similar to those of the TDC sensor itself, constituting a moderate resource usage. Compared to existing methods, our approach eliminates the need to store numerous previous sensor measurements to compute long-term metrics. Using Hamming weight to simplify the measurements also reduces the word size. These improvements greatly reduce the required input data size of our calculations. In the absence of explicit resource usage, this metric provides quantitative insights into the resource usage of existing detection schemes.

Table 4: Comparison between existing detection methods [9, 19] and this work.

Attack Detection Criterion	**Location-Agnostic**	**Max. Delay Until Detection**	**Sampling Period** (Green highlight: can detect short attacks)	**Input Data Size (bits)** (Nb. Stored Measurements × Word Size)	**Total Resource Utilization**
Static Thresholds (200 Mhz)	✗	5 ns	5 ns	**64** (64 × 1)	TDC Sensor + 8 LUTs
Variation from the average value [19]	✓	1.25 µs	1.25 µs	**3200** (400 × 8)	Not Evaluated
Accumulated Hamming Distance [9]	✓	N/A	N/A	**65536** (1024 × 64)	Not Evaluated
This work	✓	23.6 ns	5 ns	**32** (4 × 8)	TDC Sensor + 234 LUTs + 15 CARRY8 + 46 Registers + 1 DSP

6 Conclusion

This article describes the design and implementation of a novel detection scheme for voltage drop attacks on multi-tenant FPGAs. Through FPGA-based imple-

mentation results measured in real-time, we prove the efficiency of our metric based on running variance in conditions where existing countermeasures based on static thresholds would be inefficient. Unlike similar detection systems that rely on arithmetic calculations, we evaluated it against short voltage drops designed to provoke timing faults. We also eliminate the need to rely on the storage of hundreds of precedent measurements, making our approach lightweight compared to previous work. Finally, we consider the impact of noise caused by benign modules in order to determine an appropriate value for our detection threshold M and minimise the likelihood of false positives. We emphasize the complementary nature of our detection method with existing countermeasure schemes for attacker deactivation [9, 22], and to mitigate the impact of faults on the victim device [15, 21]. Because it is location-agnostic and fairly lightweight in resource usage, multiple instances of the variance-based detection module can be placed at various locations in the FPGA to infer the location of the attacker following the methods described in previous work [9, 19]. This will be considered in future work.

Further possible leads for future work, to further improve the use of our detection method in efficient countermeasures, include characterizing it in other attack scenarios. Although all existing attack circuits for voltage drop attacks produce a similar voltage drop, our detection metric requires to be studied against attacks based on memory collisions and other primitives than ROs [1]. Furthermore, recent work shows advanced attack schemes that take advantage of the resonant frequency of the PDN, and activate the attack circuits fraction per fraction, using a method called *staggering*. The effectiveness of our detection scheme against these advanced attacks is still to be evaluated, though these attacks should produce a similar voltage drop to those considered in this article. Another lead for future work is investigating the use of other voltage sensors than TDCs or ROs. Indeed, the TDC itself is the main bottleneck of most countermeasure schemes. Although its sampling period is short, it is still long compared to the time it takes for an attacker to provoke a voltage drop. The use of optimized TDC implementations [20] for attack detection will be investigated in future work.

Acknowledgments. This research is supported by the French National Research Agency (ANR) under the *CoPhyTEE* Young Researcher (JCJC) project contract ANR-23-CE39-0003-01.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Bibliography

[1] Alam, M.M., Tajik, S., Ganji, F., Tehranipoor, M., Forte, D.: RAM-Jam: Remote Temperature and Voltage Fault Attack on FPGAs using Memory Collisions. In: 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 48–55, IEEE, Atlanta, USA (Aug 2019), https://doi.org/10.1109/FDTC.2019.00015

[2] Alrahis, L., Nassar, H., Krautter, J., Gnad, D., Bauer, L., Henkel, J., Tahoori, M.: MaliGNNoma: GNN-Based Malicious Circuit Classifier for Secure Cloud FPGAs. In: 2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 383–393, IEEE, Tysons Corner, VA, USA (May 2024), ISSN 2765-8406, https://doi.org/10.1109/HOST55342.2024.10545411

[3] Ding, T.H., Li, Y.S., Jiang, D.C., Qu, Y.Z., Yan, X.: Estimation Method for Simultaneous Switching Noise in Power Delivery Network for High-Speed Digital System Design. Progress In Electromagnetic Research **125**, 79–95 (2012), ISSN 1559-8985

[4] Ebrahimabadi, M., Mehjabin, S.S., Viera, R., Guilley, S., Danger, J.L., Dutertre, J.M., Karimi, N.: Detecting Laser Fault Injection Attacks via Time-to-Digital Converter Sensors. In: 2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 97–100, IEEE, McLean, USA (Jun 2022), https://doi.org/10.1109/HOST54066.2022.9840318

[5] Gnad, D.R., Oboril, F., Kiamehr, S., Tahoori, M.B.: Analysis of transient voltage fluctuations in FPGAs. In: 2016 International Conference on Field-Programmable Technology (FPT), pp. 12–19, IEEE, Xi'An, China (Dec 2016), https://doi.org/10.1109/FPT.2016.7929182

[6] Gravellier, J., Dutertre, J.M., Teglia, Y., Loubet-Moundi, P.: High-Speed Ring Oscillator based Sensors for Remote Side-Channel Attacks on FPGAs. In: 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp. 1–8, IEEE, Cancun, Mexico (Dec 2019), ISSN 2640-0472, https://doi.org/10.1109/ReConFig48160.2019.8994789

[7] Gross, M., Krautter, J., Gnad, D., Gruber, M., Sigl, G., Tahoori, M.: FPGANeedle: Precise Remote Fault Attacks from FPGA to CPU. In: Proceedings of the 28th Asia and South Pacific Design Automation Conference, pp. 358–364, ASPDAC '23, Association for Computing Machinery, New York, NY, USA (Jan 2023), ISBN 978-1-4503-9783-4, https://doi.org/10.1145/3566097.3568352

[8] Hadipour, H.: AES-VHDL (Nov 2025), URL https://github.com/hadipourh/AES-VHDL

[9] Kajol, M.A., Sunkavilli, S., Yu, Q.: AHD-LAM: A New Mitigation Method against Voltage-Drop Attacks in Multi-tenant FPGAs. In: 2023 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp. 1–6, IEEE,

Tianjin, China (Dec 2023), https://doi.org/10.1109/AsianHOST59942.2023.10409460

[10] Krautter, J., Gnad, D.R.E., Tahoori, M.B.: FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, suitable for DFA on AES. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 44–68 (Aug 2018), ISSN 2569-2925, https://doi.org/10.13154/tches.v2018.i3.44-68

[11] La, T.M., Matas, K., Grunchevski, N., Pham, K.D., Koch, D.: FPGADefender: Malicious Self-oscillator Scanning for Xilinx UltraScale + FPGAs. ACM Trans. Reconfigurable Technol. Syst. **13**(3), 15:1–15:31 (Sep 2020), ISSN 1936-7406, https://doi.org/10.1145/3402937

[12] Le Gonidec, G., Bouffard, G., Prevotet, J.C., Méndez Real, M.: Do Not Trust Power Management: A Survey on Internal Energy-based Attacks Circumventing Trusted Execution Environments Security Properties. ACM Trans. Embed. Comput. Syst. **24**(4), 63:1–63:35 (Jul 2025), ISSN 1539-9087, https://doi.org/10.1145/3735556

[13] Le Gonidec, G., Bouffard, G., Prévotet, J.C., Méndez Real, M.: Source code repository for the variance-based detection of voltage drop attacks (2026), URL https://sourcesup.renater.fr/projects/detectionva

[14] Luo, Y., Gongye, C., Ren, S., Fei, Y., Xu, X.: Stealthy-Shutdown: Practical Remote Power Attacks in Multi - Tenant FPGAs. In: 2020 IEEE 38th International Conference on Computer Design (ICCD), pp. 545–552, IEEE, Hartford, USA (Oct 2020), ISSN 2576-6996, https://doi.org/10.1109/ICCD50377.2020.00097

[15] Luo, Y., Xu, X.: A quantitative defense framework against power attacks on multi-tenant FPGA. In: Proceedings of the 39th International Conference on Computer-Aided Design, pp. 1–9, ICCAD '20, Association for Computing Machinery, New York, NY, USA (Dec 2020), ISBN 978-1-4503-8026-3, https://doi.org/10.1145/3400302.3415694

[16] Mahmoud, D., Stojilović, M.: Timing Violation Induced Faults in Multi-Tenant FPGAs. In: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1745–1750, IEEE, Florence, Italy (Mar 2019), ISSN 1558-1101, https://doi.org/10.23919/DATE.2019.8715263

[17] Mahmoud, D.G., Dervishi, D., Hussein, S., Lenders, V., Stojilović, M.: DFAulted: Analyzing and Exploiting CPU Software Faults Caused by FPGA-Driven Undervolting Attacks. IEEE Access **10**, 134199–134216 (2022), ISSN 2169-3536, https://doi.org/10.1109/ACCESS.2022.3231753

[18] Mahmoud, D.G., Shokry, B., Lenders, V., Hu, W., Stojilović, M.: X-Attack 2.0: The Risk of Power Wasters and Satisfiability Don't-Care Hardware Trojans to Shared Cloud FPGAs. IEEE Access **12**, 8983–9011 (2024), ISSN 2169-3536, https://doi.org/10.1109/ACCESS.2024.3353134

[19] Mirzargar, S.S., Renault, G., Guerrieri, A., Stojilović, M.: Nonintrusive and Adaptive Monitoring for Locating Voltage Attacks in Virtualized FPGAs. In: 2020 International Conference on Field-Programmable Technology

(ICFPT), pp. 288–289, IEEE, Maui, USA (Dec 2020), https://doi.org/10.1109/ICFPT51103.2020.00050

[20] Moini, S., Deric, A., Li, X., Provelengios, G., Burleson, W., Tessier, R., Holcomb, D.: Voltage Sensor Implementations for Remote Power Attacks on FPGAs. ACM Transactions on Reconfigurable Technology and Systems **16**(1), 1–21 (Dec 2022), https://doi.org/10.1145/3555048

[21] Moini, S., Kansagara, D., Holcomb, D., Tessier, R.: Fault Recovery from Multi-Tenant FPGA Voltage Attacks. In: Proceedings of the Great Lakes Symposium on VLSI 2023, pp. 557–562, GLSVLSI '23, Association for Computing Machinery, New York, NY, USA (Jun 2023), ISBN 9798400701252, https://doi.org/10.1145/3583781.3590246

[22] Nassar, H., AlZughbi, H., Gnad, D.R.E., Bauer, L., Tahoori, M.B., Henkel, J.: LoopBreaker: Disabling Interconnects to Mitigate Voltage-Based Attacks in Multi-Tenant FPGAs. In: 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), pp. 1–9, IEEE, Munich, Germany (Nov 2021), ISSN 1558-2434, https://doi.org/10.1109/ICCAD51958.2021.9643485

[23] Provelengios, G., Holcomb, D., Tessier, R.: Characterizing Power Distribution Attacks in Multi-User FPGA Environments. In: 2019 29th International Conference on Field Programmable Logic and Applications (FPL), pp. 194–201, IEEE, Barcelona, Spain (Sep 2019), ISSN 1946-1488, https://doi.org/10.1109/FPL.2019.00038

[24] Provelengios, G., Holcomb, D., Tessier, R.: Power Wasting Circuits for Cloud FPGA Attacks. In: 2020 30th International Conference on Field-Programmable Logic and Applications (FPL), pp. 231–235, IEEE, Gothenburg, Sweden (Aug 2020), ISSN 1946-1488, https://doi.org/10.1109/FPL50879.2020.00046

[25] Provelengios, G., Holcomb, D., Tessier, R.: Mitigating Voltage Attacks in Multi-Tenant FPGAs. ACM Trans. Reconfigurable Technol. Syst. **14**(2), 9:1–9:24 (Jul 2021), ISSN 1936-7406, https://doi.org/10.1145/3451236

[26] Sugawara, T., Sakiyama, K., Nashimoto, S., Suzuki, D., Nagatsuka, T.: Oscillator without a combinatorial loop and its threat to FPGA in data centre. Electronics Letters **55**(11), 640–642 (2019), ISSN 1350-911X, https://doi.org/10.1049/el.2019.0163

[27] Wang, X., Niu, Y., Liu, F., Xu, Z.: When FPGA Meets Cloud: A First Look at Performance. IEEE Transactions on Cloud Computing **10**(2), 1344–1357 (Apr 2022), ISSN 2168-7161, https://doi.org/10.1109/TCC.2020.2992548

[28] Welford, B.P.: Note on a Method for Calculating Corrected Sums of Squares and Products. Technometrics **4**(3), 419–420 (Aug 1962), ISSN 0040-1706, https://doi.org/10.1080/00401706.1962.10490022