Active Fences against Voltage-based Side Channels in Multi-Tenant FPGAs

Jonas Krautter*, Dennis R.E. Gnad*, Falk Schellenberg†, Amir Moradi†, and Mehdi B. Tahoori*
*Institute of Computer Engineering, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
†Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany
*{jonas.krautter, dennis.gnad, mehdi.tahoori}@kit.edu
†{falk.schellenberg, amir.moradi}@rub.de

***Abstract*—Dynamic and partial reconfiguration together with hardware parallelism make FPGAs attractive as virtualized accelerators. However, recently it has been shown that multi-tenant FPGAs are vulnerable to remote side-channel attacks (SCA) from malicious users, allowing them to extract secret keys without a logical connection to the victim core. Typical mitigations against such attacks are *hiding* and *masking* schemes, to increase attackers' efforts in terms of side-channel measurements. However, they require significant efforts and tailoring for a specific algorithm, hardware implementation and mapping. In this paper, we show a hiding countermeasure against voltage-based SCA that can be integrated into any implementation, without requiring modifications or tailoring to the protected module. We place a properly mapped *Active Fence* of ring oscillators between victim and attacker circuit, enabled as a feedback of an FPGA-based sensor, leading to reduced side-channel leakage. Our experimental results based on a Lattice ECP5 FPGA and an AES-128 module show that two orders of magnitude more traces are needed for a successful key recovery, while no modifications to the underlying cryptographic module are necessary.**

I. Introduction

To further drive the computing efficiency in the coming years, it is increasingly interesting to virtualize FPGA resources as accelerators in SoCs or computing clouds. This promises higher computing efficiency in various workloads from machine learning to database applications. Currently, FPGAs are already being offered as a service on Amazon [4] and Alibaba Cloud [3] platforms, while being used as specific application accelerators in Microsoft Azure [30]. In addition, FPGA SoCs from Xilinx and Intel tightly integrate FPGA logic with standard ARM or x86 processor cores [27].

Particularly in cloud environments, virtualized FPGAs promise a much more efficient utilization, by allowing multi-tenant access [14], [10], [15], which also includes basic security considerations [35]. However, before widespread adoption of multi-tenant use becomes feasible, there are still critical FPGA-specific security challenges to be solved [18]. This is mainly due to some recently shown attacks that exploit the underlying electrical level of systems integrating FPGA logic [26], [34], [38], and are thus not easy to prevent [18]. These seminal attacks show that the two classical types of implementation attacks are also applicable in the virtualized FPGA scenario—with the attacker residing within one part of the FPGA fabric:

(I) Fault attacks introduce glitches during a cryptographic operation, resulting in a faulty computation. A mathematical analysis of the faulty outputs might reveal sensitive information such as secret keys [8]. Indeed, recent work demonstrated creating successful faults in neighboring FPGA-tenants by an artificially induced excessive power consumption [17], [26]. Attempts to cause faults can often be detected, i.e., by sensors or error detection [5].

(II) Power analysis side-channel attacks on the other hand measure the power or voltage fluctuations of the device to extract sensitive information [25]. Likewise, recent work showed successful side-channel attacks by placing a voltage sensor within the same FPGA-fabric without any logical connection, capturing fluctuations on the Power Distribution Network (PDN) [33], [38]. However, due to their passive and solely observing nature, such attacks cannot be detected within the system. Instead, countermeasures against power analysis attacks try to reduce the information that can be gained from measurements to a minimum, categorized in two general groups of *hiding* and *masking* countermeasures.

Masking schemes are implemented on the algorithmic level and focus on randomizing internal values to detach the power consumption from the actual secret data being processed. Especially for non-linear functions, this change comes with a large overhead. Furthermore, implementing such countermeasures is challenging and closely tied to the algorithm to be protected. Instead, hiding aims at reducing the Signal to Noise Ratio (SNR) at the electrical level, i.e., either by raising the noise floor using additional noise sources or by equalizing the instantaneous power consumption. For the latter, many schemes use some variant of dual-rail precharge logic for equalization [13]. Unfortunately, this barely works on ASICs due to manufacturing tolerances and cannot be applied directly to FPGAs. Although some recent work duplicated and inverted the whole circuit [36], the very coarse grained access to FPGA resources hinders a straightforward implementation of such schemes [32]. In practice, hiding and masking can be used side-by-side to increase the security level significantly [32].

Whereas previous work on countermeasures considers an external attacker, there have been no results against an on-chip attacker. Note that passive fences to enforce logical isolation (cf. [12], [28], [22], [35]) are entirely ineffective against attacks on the electrical level. It is also possible to check for potentially malicious design signatures, such as voltage sensors, in user bitstreams before permitting the upload to a multi-tenant FPGA [18]. However, this bitstream checking methodology must be updated continuously to account for new ways of sensor implementation. Instead, we propose an active fence on the electrical level that adaptively equalizes the power consumption that is visible to other tenants on the FPGA. Our solution uses Ring Oscillators (ROs) as an artificial source of power consumption and voltage fluctuations, whenever needed as indicated by an internal in-fabric voltage sensor. By incorporating the capabilities of an attacker to counteract the side-channel leakage on-the-fly, we achieve some form of inherent symmetry between attacker and defender. Furthermore, the countermeasure is independent of the algorithm to be protected or its implementation.

Indeed, our results show that fencing the circuit to be protected performs significantly better than placing the same amount of ROs randomly. The number of required traces can be increased even further when enabling the feedback by the internal sensor. This is the first mitigation approach that is specifically tailored against on-chip side-channel leakage in multi-tenant FPGAs.

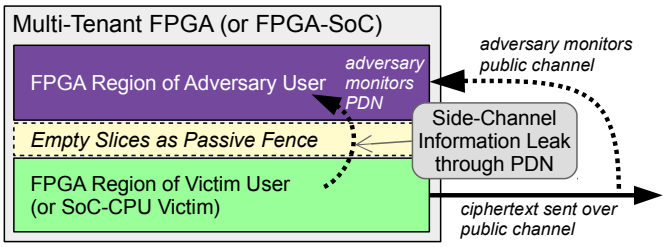


Fig. 1: Adversary model for a multi-tenant FPGA or FPGA with shared SoC Logic, such as ASIC-logic CPUs. Based on the models previously shown in [33], [38].

Overall our contributions can be summarized as:

- First mitigation method against on-chip voltage side-channel leakage in multi-tenant FPGAs.
- The hiding strategy works without changes to the cryptographic implementation and is independent of the implementation or algorithm to be protected.
- Two to three orders of magnitude leakage reduction.
- The hiding strategy can be applied to various FPGA architectures and devices from various vendors.

In the remaining paper, we first give some preliminary and background information in Section II. Furthermore, our methodology is explained in Section III, followed by our experimental setup in Section IV. Finally, we show our results in Section V and conclude the paper in Section VI.

II. Background and Related Work

A. Adversary Model

We follow the well-accepted adversary model that was previously described in various works on electrical level threats in multi-tenant FPGAs [17], [33], [26], [38], visualized in Figure 1. The adversary in this model can program an arbitrary design in a partial region of an FPGA, which is shared with at least one other victim user. There are no logic or clock signal connections between the users (i.e. tenants) of the FPGA. This isolation is ensured with empty slices put as fences between the users, as suggested by previous design practices for security in FPGAs [12], [28], [22]. Thus, the remaining connections are through the power distribution network, and all possible attack vectors are side channels. In this work, the goals of the attacker are to impact the confidentiality [33], [38], for which we present a mitigation method. Previous works also considered attacks on integrity [26] or availability [17] in the same model.

Affected systems can be SoCs or high performance computers in the cloud, utilizing FPGAs as custom accelerators. The assumed adversary uses FPGA logic to integrate a sensor to record side-channel information on the electrical level, i.e. voltage fluctuations. These voltage fluctuations can be analyzed to extract secret keys from cryptographic circuits, implemented in another part of the FPGA or SoC. In a possible real-world scenario, a victim user encrypts plaintext messages using any cryptographic core in a part of the FPGA, and sends out the encrypted ciphertext messages over a public communication channel. The adversarial user which monitors the public channel can then use the measured voltage fluctuations together with the ciphertext to extract the secret encryption key, and subsequently sensitive encrypted information.

B. On-Chip Power Distribution Networks

PDNs are designed to deliver power from a main power source through a Voltage Regulator Module (VRM) and board-level components, down to the individual transistors inside the chip [19], [7]. A stable supply voltage level (V_{dd}) inside the chip is required, since it directly affects transistor delay $\tau_d \propto 1/V_{dd}$. This delay must be below a required limit to meet physical timing constraints, s.t. $\tau_d < \tau_{constrs}$, or there may be timing violations that eventually result in errors.

The chip-level PDN consists of metal wires, which also behave as parasitic resistors (R) and inductors (L) [29], [19]. Workload and data-dependent switching activity, toggling at f_{sw}, results in a changing electrical current flow in time $I(t) \propto f_{sw}$. Dependent on the parasitic components, this affects the voltage level, separated into a resistance-dependent part $\Delta V_{drop,R} = I(t)R$, and an inductive part that depends on the amount of current change over time: $\Delta V_{drop,L} = LdI(t)/dt$. In modern technologies below 45nm, the inductive voltage drop is higher than the resistive one [29], [37]. Thus, decoupling capacitors (C) (*decaps*) are added in the physical layout design stages to reduce voltage drops during operation [29], [7]. However, there is a certain level of noise that can be observed as tempo-spatial variations across the entire PDN [19], [7], [16].

As a result, we can leverage this knowledge to either cause timing violations or observe switching activity. Increasing f_{sw} leads to a high $I(t)$, which reduces V_{dd} such that τ_d is elevated above the level of $\tau_{constrs}$, causing timing violations [26]. On the other hand, long paths that purposefully do not meet physical timing requirements can be used to observe voltage dependent switching activity of other modules inside the chip [39].

C. Electrical-Level Attacks

Side-channel attacks exploit unintentional information channels to reveal sensitive internal data. For power analysis, an attacker measures the power consumption of the device or its electromagnetic emanation while it performs a cryptographic operation [25]. For example, the captured power consumption might be linked to an internal value that depends on a part of the secret key and the input. Then, the attacker can perform a statistical evaluation (c.f., [25], [9], [6]) to test which key hypothesis results in intermediate values that indeed correspond to the observed power consumption. For example for Correlation Power Analysis (CPA) [9], the correct key candidate will show the highest correlation when the attack is successful. Because the side-channel leakage is usually very small, many traces have to be recorded and evaluated. Thus, side-channel resilience is usually measured in the number of traces required for a successful attack.

Such attacks are widely studied and typically require physical access to the device to capture the power consumption, i.e., to connect the probe of an oscilloscope or to place an EM probe in the near vicinity of the device. Yet, recent work has shown remote attacks on multi-tenant FPGAs that break with this assumption, i.e., without any physical access to the device. Using the on-chip power distribution network as source or carrier of side-channel information, user-programmable FPGA primitives can be utilized in order to implement an on-chip voltage sensor [39]. Indeed, they were proven to be very effective at launching attacks on other parts of the chip [33], [38].

The underlying working principle was already introduced in the previous section: A circuit's activity will result in voltage fluctuations across the PDN, independent of any logical connection. Given that the propagation delay of a circuit depends on the supply voltage, capturing the activity of a circuit becomes the task of measuring the speed of the circuit. Figure 2 depicts two basic concepts to measure voltage in FPGAs. The straight-forward solution is to count the oscillations of a Ring Oscillator (RO), shown on the left side.

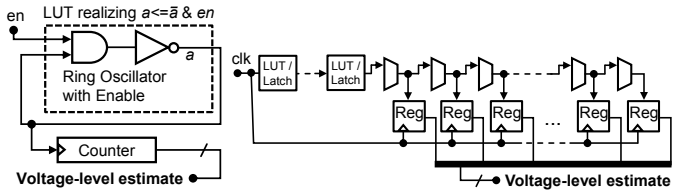


Fig. 2: Two types of voltage side-channel sensors in an FPGA as shown in [18]. Left: RO counter, right: tapped delay line.

However, implementing an RO on an FPGA usually results in a long feedback path and thus, poor resolution. A more effective approach is thus shown on the right, which measures how far a signal traveled through a delay line, and for which the clock is used in this example. This can be implemented very efficiently in FPGAs which results in outperforming sensors [39].

Another class of implementation attacks are fault attacks to extract sensitive information [8]. Many of such attacks are based on a mathematical evaluation of the output of a correct and a faulty computation. Although protecting against such attacks is not within the scope of this work, we will later rely on circuits that were previously considered to be an offensive tool: The excessive power consumption of highly active elements such as ROs can be used to successfully create computation errors in neighboring circuits residing within the same FPGA fabric [26].

D. Side-Channel Attack Countermeasures

There is ongoing research to explore potential countermeasures against such attacks, following the two directions of *masking* or *hiding*: Masking schemes are implemented on the algorithmic level and separate internal values into multiple shares [23]. Only the combination of all shares will again reveal the correct secret value. Ideally, the operations are performed individually on each share. Since this is not possible for non-linear functions, ensuring the correctness of the computation requires a large algorithmic overhead. Furthermore, such a protection has to be developed individually for each algorithm to be protected and its actual implementation is challenging as well. Masking techniques can be attacked with higher order attacks, i.e., using higher statistical moments [31]. However, for such attacks, the noise is amplified exponentially with increasing order, resulting in an increased number of required traces.

On the other hand, hiding aims at reducing the SNR at the electrical level. One option is to increase the noise level through additional sources of noise. Especially on FPGAs, implementing generic noise generators has been studied using shift registers, BRAM write collision, or short-term short circuits [20]. Correlated noise generation was suggested to hide leakage from an FPGA AES implementation in [24]. Likewise, [20] proposes clock randomization to temporally spread the side-channel information. Another variant of hiding reduces the strength of the signal by equalizing the instantaneous power consumption. In the past, hiding was often implemented using a variant of dual-rail precharge logic [13]. While in theory, this results in a balanced power consumption, perfectly balanced computation paths are hard to achieve in practice due to manufacturing process variations. This problem is amplified on standard FPGAs as the physical realization of elements is unknown. Some recent work proposed duplicating and inverting a cryptographic circuit [36]. Yet, this is still difficult to achieve on FPGAs, since on-chip resources can only be used very coarse grained [32]. More lightweight and feasible approaches achieve power equalization on FPGAs to some limited extent using ROs. However, previous approaches were not designed with side-channel attacks in mind and are thus not evaluated in a security context so far [21].

Summarizing the existing solutions, they are often specific to a particular cryptographic algorithm and require significant changes to its implementation. Our active fence aims to overcome this disadvantage but can also be applied in addition to any of the more sophisticated, specific countermeasures, to further increase protection against side-channel attacks. In Section II we explained how the knowledge of on-chip power delivery is used to measure and cause voltage fluctuations in digital logic, reproducible with user-programmable FPGA primitives. On-chip power analysis side-channel attacks are based on the fact that a cryptographic module injects voltage fluctuations into the power grid, depending on its data-dependent activity. These fluctuations travel very quickly through the on-chip power grid [7], [16], and reach other parts of the chip. The effect they have on path delays can then be observed by an attacker using the delay-line sensors described in Section II.

The same underlying mechanisms can be used to add additional voltage fluctuations into the system, to change what can be observed at the side of the adversary. Our proposed methods try to reduce the overall usable SNR in the voltage fluctuations traveling from the cryptographic victim module to the adversarial sensor by putting active fences between the modules. We base these active fences on ROs, as controlled primitives, that can be enabled and disabled. During the time they are enabled, they have a high switching activity f_{sw} that leads to a high current $I(t)$. Thus, with only small area use they can efficiently inject a high voltage drop into the power grid to equalize the outgoing voltage fluctuations (i.e., the information leakage) from the cryptographic module. Just like other countermeasures from the *hiding*-category, we evaluate these active fences to either increase the noise, or reduce the signal. Overall, this is evaluated based on the number of traces required for a successful CPA attack.

In Figure 3, we present a general overview of the proposed method. The active fence is placed as a RO region around a design, that is critical to security of an application, to prevent side-channel leakage to partitions of other users. Various strategies of placing and activating the ROs within the fence region are feasible. ROs can be activated randomly or by using a Time-to-Digital Converter (TDC) sensor for controlled activation. We will discuss those strategies in the following subsections and show experimental results for some combinations.

III. Methodology

A. Placing Ring Oscillators

To equalize the voltage fluctuations from the cryptographic module, various choices to place ROs are possible. For external power analysis, an arbitrary placement of ROs may be sufficient. However, due to the strong tempo-spatial dependency of on-chip voltage fluctuations [7], [16], it is important how the ROs are placed, and in which order they get activated. This plays an important role in suppressing the side-channel leakage an adversary can observe.

Additionally, we should consider reducing the impact this countermeasure has on all benign users of the multi-tenant FPGA, i.e. ideally it should be placed outside of the user-assigned reconfigurable regions. With all these considerations, our choices can be broken down to two basic options:

1) Constrain ROs to a region between attacker and victim in which the place-and-route toolchain places them heuristically.
2) Map the ROs precisely into a densely packed uniform array between attacker and victim, to activate them in a specific spatial order.

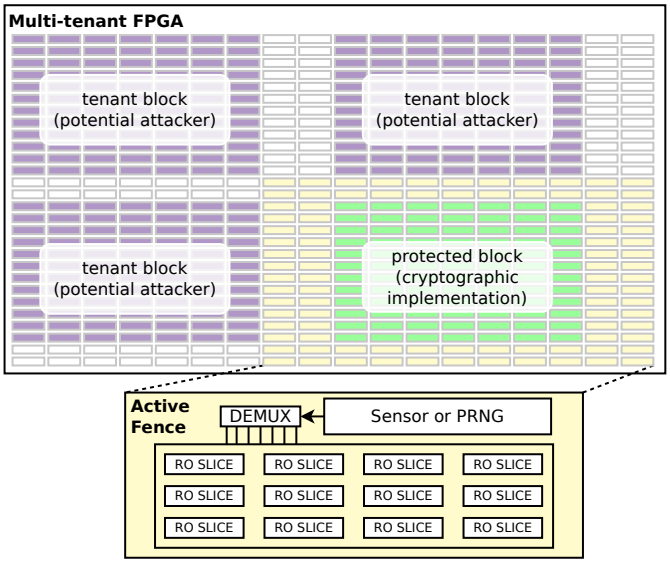


Fig. 3: Overview of the active fence as a protective countermeasure between a design, which is used in a security-related application, and other users on a multi-tenant FPGA. A specific amount of ROs is activated dependent on the value of a sensor or a Pseudo-Random Number Generator (PRNG).

When placing the RO array heuristically into the constrained region, activation happens at arbitrary locations within that region. For the second option, the ROs are activated row-by-row or column-by-column, depending on the layout of the entire system.

In Figure 4 we show a simplified example of the entire design when allowing the place-and-route toolchain placing ROs heuristically. Depending on the input value, we activate a larger or smaller amount of ROs in the active fence region to mitigate side-channel leakage from the victim partition at the bottom of the figures into the attacker region at the top. Figure 5 outlines the principle of a constrained row-by-row RO placement and activation.

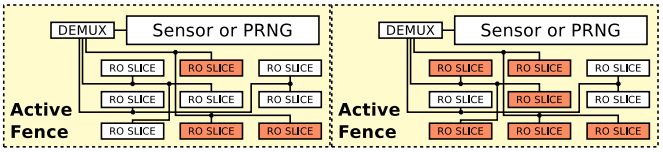
B. Activation Strategies

In addition to the placement layout of the ROs regarding victim and adversarial blocks, it is also important to decide on an activation strategy. Generally, we intend to lower the supply voltage when activating a larger amount of ROs and raise the voltage when activating a smaller amount. The exact amount is technology dependent and can be experimentally evaluated, such that enabling all ROs can sufficiently out-balance the worst-case voltage fluctuations emitted from the cryptographic module. It must be decided how these ROs will be activated at runtime — both spatially and temporally. An activation strategy can fulfill one or both goals of either canceling out the fluctuations caused by the cryptographic module, or to add overall more noise to the system.

In our experiments we consider two strategies as depicted also in Figure 3:

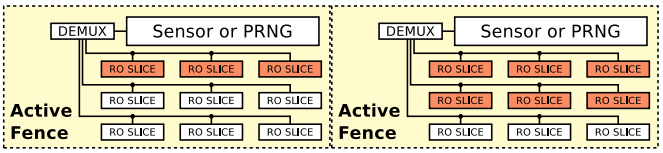
1) Random activation patterns, where the amount of activated ROs is based on a PRNG output, implemented as a Linear Feedback Shift Register (LFSR).
2) Activating an amount of ROs depending on the value of a voltage-fluctuation sensor, similar to the sensor used by the attacker, however at the victim side.

A random activation fulfills the objective of increasing the noise in the system, increasing the amount of required traces for a successful attack. When activating according to a sensor value, we aim to flatten



(a) One third of ROs activated (b) Two thirds of ROs activated

Fig. 4: RO activation pattern when letting the toolchain decide the placement heuristically



(a) One third of ROs activated (b) Two thirds of ROs activated

Fig. 5: RO activation pattern when fixing RO placement as a row-by-row grid

the voltage fluctuations caused by the cryptographic module, thus reducing the leakage to the attacker as well. In our experiments, we find a sensor-based activation to be slightly more successful in weakening the attack.

IV. Implementation and Experimental Setup

Our experimental setup is based on the open-source FPGA development board *Radiona ULX3S* [1] in a version that integrates a Lattice ECP5 12F FPGA with 12K LUT elements with various other components such as an Espressif ESP32 IoT microcontroller module. Although Xilinx and Intel FPGAs are used more widely, we choose to initially develop our design on a Lattice device, which has great open-source support [2] that will be useful in future research. Nevertheless, our proposed methodology is assumed to be easily adaptable to other FPGAs and platforms.

Figure 6 shows the overview block diagram of the setup. The Lattice ECP5 on the left side contains three modules. In the bottom part, a hardware AES module is integrated, which is connected to the outside to receive plaintexts, encrypt them, and send the ciphertext back to the connected workstation PC. This module resembles the victim user, out of two users of the system. For the connection, we use a simple UART module on the FPGA, connected to a USB-to-UART Bridge on the Radiona ULX3S Board. In the top part of the ECP5, the adversary user implements his logic, based on a delay-line sensor, specifically tailored to the ECP5 primitives, as we describe in Subsection IV-B. Between the two users, we implement the different variants of our active fence, as described in Section III. Both the AES module and the active fence to protect it are clocked from the same on-chip Phase Locked Loop (PLL) clock generator, whereas the adversary uses another PLL. On the board level, both attacker and victim PLLs are connected to the same onboard 25 MHz clock generator.

In Figure 7 we show how the design is mapped onto the ECP5 FPGA as seen in the Floorplan View of the Lattice Diamond design software. To make our results comparable, we fix the placement of the AES module as well as the attacker TDC sensor. This experimental setup follows roughly what has been presented in [33] and [38], except for the added active fences.

A. AES implementation

In this paper we use industry's standard symmetric encryption scheme AES in its 128-bit variant to evaluate the performance of

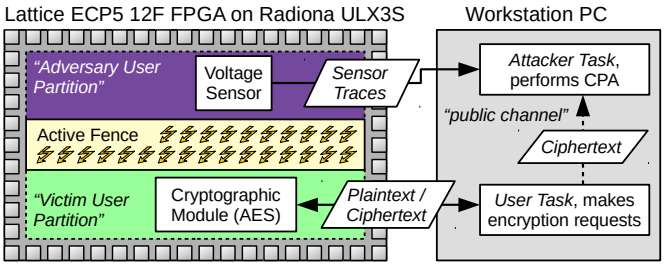


Fig. 6: Experimental setup overview showing the Lattice ECP5 FPGA connected to a workstation PC. The AES module is used to encrypt messages, while an attacker can use the ciphertexts together with sensor traces to perform CPA.

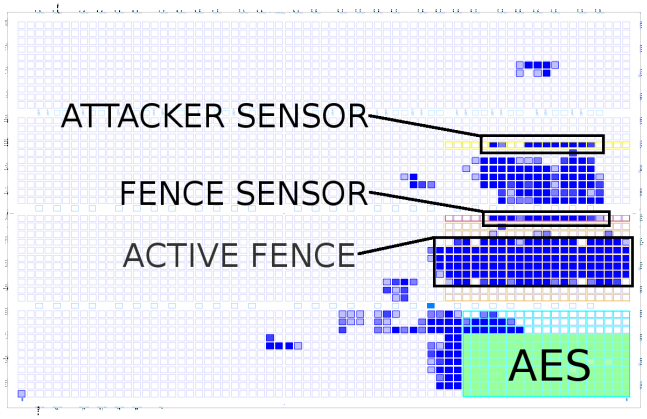


Fig. 7: The complete layout of the design on the ECP5 FPGA as seen in the Lattice Diamond Floorplan Viewer

our protection against side-channel analysis. However, note that our proposed active fence is independent of the exact algorithm or implementation to be protected. To keep comparability, we use an AES module that follows the same design principle as explained in [33], using a 32-bit datapath. In each clock cycle, four of the 16 state bytes are processed in parallel, as visualized in Figure 8. One round thus takes four clock cycles and one additional clock cycle to perform the key schedule sharing the Sboxes (not depicted). The module is operated at a clock speed of 12.5 MHz from the same clock generator as the active fence.

B. Sensor Implementation in Lattice ECP5

Previous publications have mostly used Xilinx FPGAs [33], [38], [34], but it was also reported that the attack was successful on a Lattice iCE40, without providing any further details on the sensor construction [18]. We implement TDC sensors as explained in [18] for both offensive and defensive design parts on the Lattice ECP5, using the available carry-chain primitives.

Carry-chains on the ECP5 are instantiated through *CCU2C* primitives, each of which represents a 2-bit carry element with carry-in, carry-out and two bits of summation signals. Connecting the carry-in of one *CCU2C* element to the carry-out of another element automatically enforces an adjacent placement of the two elements.

We route the sensor clock signal through an initial delay carry-chain into the main chain, which is eventually sampled into registers at the falling sensor clock edge. A specific location is set for the first elements of the initial and the main carry-chain respectively, which fixes the placement of the entire sensor due to the described adjacent placement of *CCU2C* elements. Although the initial delay

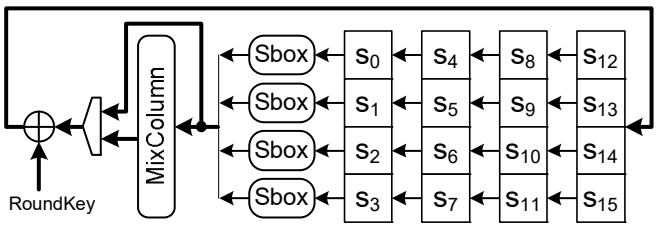


Fig. 8: Schematic of the used AES encryption core design as shown in [33]

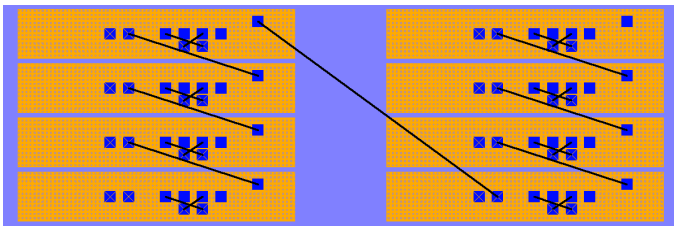


Fig. 9: Partial sensor carry-chain on the ECP5 as displayed in the Floorplan View of the Lattice Diamond design software; Each block contains a 2-bit carry element and the connection between those elements enforces adjacent placement as a fast carry chain on the ECP5

length varies depending on the sensor clock, we fix the output carry-chain at a length of 62. When the supply voltage is high, the sensor clock signal propagates into the delay line further than at low voltage until the falling edge causes the sensor registers to capture the state of the line. We count the number of set bits in the main delay line as the sensor output, which gives us a 6-bit sensor value.

In Figure 9, we show 16 bits of a TDC delay line on the ECP5, as seen in the Floorplan View of the Lattice Diamond design software. For our experiments we sample at 100 MHz on the attacker side sensor, whereas the TDC sensor on the Active Fence part was restricted to 50 MHz to account for a realistic scenario. We assume that in general the attacker is able to sample the side-channel information at a higher rate compared to the clock speed of the attacked module. Thus, the attacked AES module is driven at 12.5 MHz clock speed, giving the attacker 8 samples per AES clock. As shown in the next section, the amount of traces required to attack an unprotected design is rather small.

C. Active Fence Implementation

In Section III, we explained the general principle of an active fence as a hiding countermeasure. The implementation on the Lattice ECP5 FPGA is based on single Look-Up Table (LUT) ROs, which are instantiated using the *ND2* primitive. A *ND2* element represents a two-input NAND-gate to realize a RO with an enable signal. To prevent any synthesis optimizations from removing the RO array, we combine all RO signals with an XOR operation into a single signal, which is mapped to an open pin on the FPGA.

The ROs are placed according to one of the placement strategies described in Section III and a certain amount is activated depending on a value x. We use a total amount of $21 \times 32 = 672$ ROs, where for each increase in x an additional amount of 21 ROs is activated. This total amount of ROs is chosen to correspond roughly to the amount of slices occupied by the AES module for the fence to be able to generate voltage fluctuations in the range of the cryptographic implementation itself. Furthermore, a row of 21 adjacent ROs is required to spatially cover the AES module when using a row-by-row RO layout. 32 corresponds to the maximum fluctuation range in

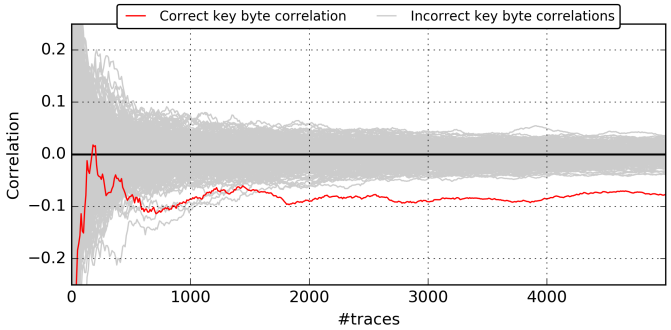


Fig. 10: Baseline results for CPA on AES without any countermeasure. The correlation for the correct key is marked red and the attack is successful after about 1800 traces.

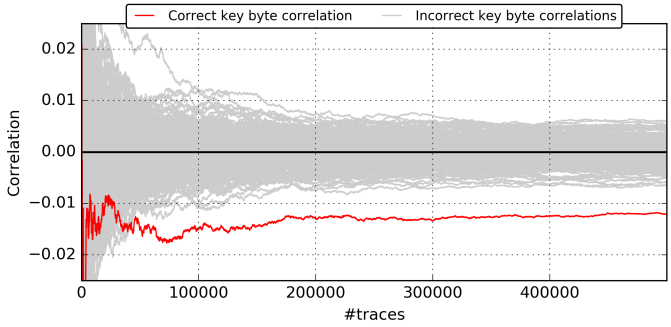


Fig. 11: Results for CPA on AES with an arbitrarily placed active fence, activated based on a PRNG output. The correlation for the correct key is marked red and the attack is successful after about 80k traces.

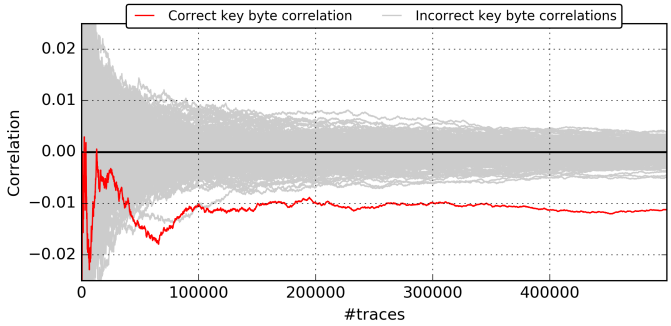


Fig. 12: Results for CPA on AES with an active fence, that is placed row-by-row and activated based on a PRNG output. The correlation for the correct key is marked red and the attack is successful after about 120k traces.

the TDC sensor we observed in our experiments. Thus, a value of $x = 20$ activates a total amount of 420 ROs.

Overall, the overhead of the active fence in terms of area is about the same as the area occupied by the AES module. Analyzing the required additional power consumption by the design with active fence as reported by the Lattice Diamond Power Calculator is 178 μW. This corresponds to slightly more than half of the additional power consumption reported for the AES module, which is 320 μW. However, the tool does most likely not account for the dynamic power consumption of the ROs adequately and thus further measurements would be required.

In Subsection III-B we explained different activation methods. We can choose x either as a PRNG output or a TDC sensor value. For the sensor-based activation, we scale the sensor value into the range of 0 to 32 by subtracting the observed minimum in the first encryptions. To randomly activate the RO grid, we simply use x directly as the output of a PRNG module from OpenCores [11].

V. Results

Our results are based on performing a CPA on an AES encryption module in different scenarios using the voltage traces collected through the attacker TDC sensor. For each scenario, we deploy variations of our active fence implementation or omit any protection for the baseline experiment. The success of any defensive method is evaluated as the increase in traces required for the attacker to successfully recover the first secret key byte of the last AES round key. For CPA we use a Hamming distance model and compute bitwise correlation as described in [33]. After selecting the best correlating bit, we plot the key hypothesis correlations over the number of evaluated traces with the correct key marked red to show the minimum traces required for a successful attack.

A. Baseline, without mitigation

Initially, we synthesize a design without any countermeasures, to compare the efficiency of our active fence countermeasure against. In Figure 10, the result of a CPA on the unprotected AES is shown as described previously. We see that the attacker is able to successfully distinguish the correct key byte hypothesis after about 1800 traces. Comparing these results to those reported before in [33], [38], [18], we need a similar amount of traces as Schellenberg et al.

B. Area-restricted ROs, randomly enabled

For a first experiment, we naively let the design tool place ROs into a constrained area between victim and attacker partitions and randomly activate different amounts of them. The amount of activated ROs is determined by the output of an LFSR-based PRNG. Figure 11 shows the result of a CPA on AES in this scenario. We see that the amount of traces required for a successful attack increases from 1800 to 80k. In general, that makes this approach already a valid defense mechanism, but we show in further experiments, that we are able to improve the defense by specific placement and activation.

C. RO-array, randomly enabled

The hiding effect becomes much stronger, when placing ROs in a specific manner, to prevent the attacker from profiting from placement convenience. As described in Section III and Section IV, we place ROs in a row-by-row scheme between AES module and attacker sensor. Again using a PRNG output to determine the amount of activated ROs, we show in Figure 12 how CPA performs in this scenario.

The correlation with the correct key byte is now much lower in absolute value. A successful attack is possible after 120k traces, where the absolute correlation with the correct key is larger than the correlations with the incorrect keys. In our last experiment, we show how the attack success can be decreased more reliably with a sensor-based active fence.

D. RO-array, sensor-based enabled

Lastly, we determine the amount of ROs to be activated using the output value of a TDC sensor, clocked at half the speed of the sensor used by the attacker. The goal of this activation strategy is to directly mitigate the voltage fluctuations caused by the AES encryption. In Figure 13, we show the results of a CPA on the AES module in this setup. As in the previous experiment, the absolute correlation value decreases significantly. Additionally, the attack is only successful after about 300k traces, which corresponds to two magnitudes of

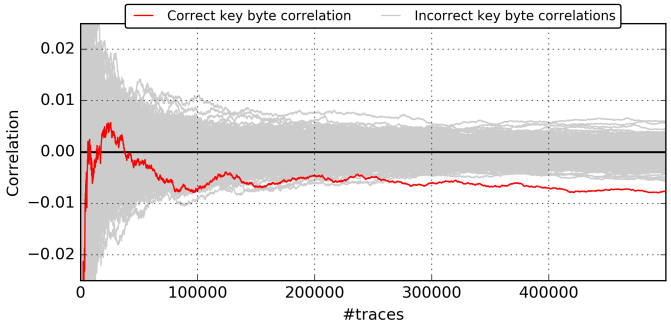


Fig. 13: Results for CPA on AES with a sensor-based active fence, that is placed row-by-row. The correlation for the correct key is marked red and the attack is successful after about 300k traces.

additional traces being required when compared to attacks on the unprotected module. Further improvements to the sensor feedback will most certainly lead to an even better leakage mitigation in future works.

E. Discussion

We developed a first approach for an on-chip hiding countermeasure against side-channel attacks on multi-tenant FPGAs. In this scenario, the limitations of the attackers are known, whereas an external attacker can employ any expensive measurement equipment. The evaluation of different placement and activation strategies shows that the general approach of active fences between different designs on a multi-tenant FPGA is feasible. However, to significantly improve the hiding effect further investigation of the spatial dependencies and improvements of the defensive sensor are certainly required.

Currently, we employ a clocked sensor, similar to the attacker sensor to mitigate fluctuations caused by an AES module. A better approach would be an entirely combinational circuit to enable an almost instantaneous reaction of the active fence.

The benefit of the presented method lies mostly within its generality and simplicity as leakage can be prevented independently of the underlying cryptographic module, implementation, and mapping.

VI. Conclusion

In this paper, we could show how a specifically placed and activated active fence made out of ring oscillators can effectively reduce the on-chip side-channel leakage by two orders of magnitude. Additionally, the approach is simple to deploy and independent of the cryptographic module that should be protected. In future works, the specific implementation may be improved, for example through more advanced on-chip sensors which can react to voltage fluctuations instantaneously. Moreover, the connection between placement of the fence elements and success of the countermeasure should be investigated further.

Hiding schemes like what we propose in this work are known to be unable to prevent side-channel attacks entirely. Instead, they make the attacks harder to mount, i.e., a higher number of measurements is required. Therefore, masked implementations can benefit more from additionally integrated hiding countermeasures.

Hence, examining the effect of our developed on-chip hiding technique on higher-order side-channel attacks when the underlying cryptographic module is a masked implementation is among our future plans. Our method is also orthogonal to the approach of checking bitstreams for malicious signatures before uploading them to a multi-tenant FPGA which has been proposed previously as a countermeasure to security threats in multi-tenant FPGAs.

References

[1] (2019) Radiona ULX3S. Radiona.org / Zagreb Makerspace and FER – Faculty of Electrical Engineering and Computing – University of Zagreb. [Online]. Available: https://radiona.org/ulx3s/

[2] (2019) SymbiFlow – open source FPGA tooling for rapid innovation. [Online]. Available: https://symbiflow.github.io/

[3] Alibaba Cloud. (2018) Deep dive into alibaba cloud F3 FPGA as a service instances. [Online]. Available: https://www.alibabacloud.com/blog/deep-dive-into-alibaba-cloud-f3-fpga-as-a-service-instances_594057

[4] Amazon. (2019) EC2 F1 instances. [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/

[5] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, 2006.

[6] L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon, "Mutual information analysis: a comprehensive study," *Journal of Cryptology*, vol. 24, no. 2, pp. 269–291, 2010.

[7] R. Bertran, A. Buyuktosunoglu, P. Bose, T. J. Slegel, G. Salem, S. Carey, R. F. Rizzolo, and T. Strach, "Voltage noise in multi-core processors: Empirical characterization and optimization opportunities," in *47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2014, pp. 368–380.

[8] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Advances in Cryptology — EUROCRYPT*. Springer, 1997, pp. 37–51.

[9] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Lecture Notes in Computer Science (LNCS)*. Springer, 2004, pp. 16–29.

[10] S. Byma, J. G. Steffan, H. Bannazadeh, A. L. Garcia, and P. Chow, "FPGAs in the Cloud: Booting Virtualized Hardware Accelerators with Openstack," in *International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2014, pp. 109–116.

[11] J. Castillo Villar. (2019) Systemc/verilog random number generator. OpenCores. [Online]. Available: https://opencores.org/projects/systemc_rng

[12] J. D. Corbett, "The Xilinx Isolation Design Flow for Fault-Tolerant Systems," 2013.

[13] J.-L. Danger, S. Guilley, S. Bhasin, and M. Nassar, "Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors," in *3rd International Conference on Signals, Circuits and Systems (SCS)*. IEEE, 2009.

[14] K. Eguro and R. Venkatesan, "FPGAs for trusted cloud computing," in *International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2012, pp. 63–70.

[15] S. A. Fahmy, K. Vipin, and S. Shreejith, "Virtualized FPGA Accelerators for Efficient Cloud Computing," in *CloudCom*. IEEE, 2015, pp. 430–435.

[16] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "An experimental evaluation and analysis of transient voltage fluctuations in fpgas," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 99, pp. 1–14, 2018.

[17] D. R. E. Gnad, F. Oboril, and M. B. Tahoori, "Voltage Drop-based Fault Attacks on FPGAs using Valid Bitstreams," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 4–8.

[18] D. R. E. Gnad, S. Rapp, J. Krautter, and M. B. Tahoori, "Checking for electrical level security threats in bitstreams for multi-tenant FPGAs," in *International Conference on Field-Programmable Technology (FPT)*. IEEE, 2018.

[19] M. S. Gupta, J. L. Oatley, R. Joseph, G.-Y. Wei, and D. M. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2007, pp. 1–6.

[20] T. Güneysu and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *Cryptographic Hardware and Embedded Systems – CHES 2011*. Springer Berlin Heidelberg, 2011, pp. 33–48.

[21] H. Homulle, S. Visser, B. Patra, and E. Charbon, "Design techniques for a stable operation of cryogenic field-programmable gate arrays," *Review of Scientific Instruments*, vol. 89, no. 1, p. 014703, 2018.

[22] T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. Levin, T. Nguyen, and C. Irvine, "Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems," in *IEEE Symposium on Security and Privacy (SP '07)*. IEEE, 2007.

[23] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology - CRYPTO 2003*. Springer, 2003, pp. 463–481.

[24] N. Kamoun, L. Bossuet, and A. Ghazel, "Correlated power noise generator as a low cost DPA countermeasures to secure hardware AES cipher," in *2009 3rd International Conference on Signals, Circuits and Systems (SCS)*. IEEE, 2009.

[25] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*. Springer, 1999, pp. 388–397.

[26] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "FPGAhammer: remote voltage fault attacks on shared FPGAs, suitable for DFA on AES," *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, no. 3, 2018.

[27] "FPGA device feature list (DFL) device drivers," Linux Weekly News, Jun. 2018. [Online]. Available: https://lwn.net/Articles/757283/

[28] M. McLean and J. Moore, "FPGA-based single chip cryptographic solution," *Military Embedded Systems*, pp. 34–37, 2007.

[29] A. V. Mezhiba and E. G. Friedman, "Scaling trends of on-chip power distribution noise," *Trans. VLSI Syst.*, vol. 12, no. 4, pp. 386–394, April 2004.

[30] Microsoft. (2017) Microsoft unveils project brainwave for real-time AI. [Online]. Available: https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/

[31] A. Moradi and F.-X. Standaert, "Moments-correlating DPA," in *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*. ACM, 2016, pp. 5–15.

[32] A. Moradi and A. Wild, "Assessment of hiding the higher-order leakages in hardware," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 453–474.

[33] F. Schellenberg, D. Gnad, A. Moradi, and M. B. Tahoori, "An Inside Job: Remote Power Analysis Attacks on FPGAs," in *Design, Automation & Test in Europe (DATE)*, 2018.

[34] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "Remote inter-chip power analysis side-channel attacks at board-level," in *Proceedings of the International Conference on Computer-Aided Design - ICCAD '18*. ACM Press, 2018.

[35] S. Trimberger and S. McNeil, "Security of FPGAs in data centers," in *IEEE 2nd International Verification and Security Workshop (IVSW)*. IEEE, 2017.

[36] A. Wild, A. Moradi, and T. Güneysu, "Glifred: Glitch-free duplication towards power-equalized circuits on FPGAs," *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 375–387, 2018.

[37] R. Zhang, K. Wang, B. H. Meyer, M. R. Stan, and K. Skadron, "Architecture implications of pads as a scarce resource," in *International Symposium on Computer Architecture (ISCA)*, 2014, pp. 373–384.

[38] M. Zhao and G. E. Suh, "FPGA-Based Remote Power Side-Channel Attacks," in *Symposium on Security and Privacy (S&P)*. IEEE, 2018.

[39] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing Nanosecond-scale Voltage Attacks and Natural Transients in FPGAs," in *International Symposium on Field Programmable Gate Arrays (FPGA)*, 2013, pp. 101–104.