LeakyDSP: Exploiting Digital Signal Processing Blocks to Sense Voltage Fluctuations in FPGAs

Xin Zhang[13†], Jiajun Zou[1†], Yi Yang[1], Qingni Shen[1*], Zhi Zhang[2*], Yansong Gao[2], Zhonghai Wu[1], Trevor Carlson[3]
[1]School of Software and Microelectronics, Peking University
[2]The University of Western Australia [3]National University of Singapore
Email: {zhangxin00, jjzou2002, tkike}@stu.pku.edu.cn, qingnishen@ss.pku.edu.cn,
{zhi.zhang, garrison.gao}@uwa.edu.au, wuzh@pku.edu.cn, tcarlson@comp.nus.edu.sg

***Abstract*—In recent years, cloud providers are dedicated to enabling FPGA multi-tenancy to improve resource utilization, but this new sharing model introduces power side-channel threats, where attackers detect voltage fluctuations from co-located circuits. This paper proposes LeakyDSP, a novel on-chip sensor that maliciously configures DSP blocks to sense fine-grained voltage fluctuations but is overlooked by existing studies. Our experimental results show that LeakyDSP achieves high sensitivity to voltage fluctuations and strong robustness to different placements. Besides, we apply LeakyDSP to extract full AES keys with 25 k-78 k traces and build covert channels with a high transmission rate of 247.94 bit/s.**

I. Introduction

In recent years, leading commercial cloud service providers such as Amazon AWS [4], Alibaba [2], and Microsoft Azure [5] have introduced remote access to high-performance Field Programmable Gate Arrays (FPGAs) [7]. To maximize FPGA utilization efficiency, both academia and industry have focused their efforts on enabling the co-residence of FPGA circuits by independent tenants, referred to as multi-tenant FPGAs. For example, researchers have proposed different end-to-end frameworks to virtualize FPGA resources in the cloud environment [16], [19], [21], [22], [34], [39]. Besides, Xilinx has integrated Stacked Silicon Interconnect [37] and Dynamic Function eXchange [36] into advanced FPGAs to facilitate multi-tenant FPGAs.

While this new sharing model has attracted considerable attention from both academia and industry, it introduces new security risks, notably power side-channel leakages [14], [40], [43]. Specifically, different users' circuits on a multi-tenant FPGA share the same power delivery network (PDN) [7]. Consequently, transient voltage fluctuations in the shared PDN, induced by computing activities of a victim circuit, can be sensed by a co-located attack circuit. Leveraging this capability to sense voltage fluctuations, various remote power side channel attacks have been proposed to extract cryptographic keys [15], [33], [43], build covert channels [43], fingerprint computation workloads [14], and steal DNN model architectures [42] or inputs [25].

[†] Co-first authors. [*] Corresponding authors.

To sense voltage fluctuations, prior studies have proposed various malicious circuit designs that exploit combinations of traditional FPGA logic resources, i.e., Look-up Tables (LUTs) [15], [32], [33], [43], carry chains (CARRY) [11], routing resources [29], and flip-flops (FFs). For instance, time-to-digital converters (TDCs) [11] chain CARRY resources as a long delay line and connect each CARRY output precisely to an FF within the same slice. Besides, the ring oscillator (RO) circuits configure LUTs to implement *combinational loop* that generates oscillating signals, which are then captured by FFs.

In response, several countermeasures have been proposed to prevent the deployment of such circuits on multi-tenant FPGAs [11], [28], [31]. First, as TDC circuits require a long delay chain placed in continuous vertical areas, they can be mitigated by restricting access to such areas [11]. Second, as RO circuits inherently contain combinational loops and some TDC circuits [13] include latch structures, bitstream checks targeting these key structures have been introduced [28], [31]. Although there are new sensor designs [15], [29], [32], [33] that bypass these placement constraints and bitstream checks, all these designs are crafted by configuring traditional FPGA logic resources that are in the primary focus of bitstream checks [18]. Outside the scope of traditional FPGA resources, the DSP blocks dedicated to complex mathematical operations remain unexplored. To this end, we are interested in the following research questions:

Can voltage fluctuations be sensed through maliciously configuring DSP blocks? If so, what attacks can be launched, and what information might be leaked?

In this paper, we propose LeakyDSP, a new on-chip sensor that is built by configuring DSP blocks to sense fine-grained voltage fluctuations without requiring any traditional FPGA logic resources. In a multi-tenant FPGA environment, DSP blocks are partitioned into separate virtual areas, allowing each tenant exclusive access to their DSP blocks [22]. Our key observation is that each dedicated DSP block has three main sub-components, i.e., a pre-adder, a multiplier, and an arithmetic logic unit (ALU). The signal delay incurred by either sub-components is sensitive to voltage, resulting in different signal propagation delay between output and input of a DSP block when different voltage fluctuations occur.

To this end, we craft LeakyDSP as follows. *First*, we amplify the effect of voltage variations on the output of DSP blocks by maximizing the signal propagation delay. This is achieved by creating a chain within a single DSP block that links the aforementioned sub-components without intermediate registers. Then, multiple configured DSP blocks are connected in series to accumulate their signal delays. To capture and store the resulting flipped bits, the output register of the final DSP block is instantiated. *Last*, we employ the `IDELAY` primitive to dynamically adjust the initial delay of DSP blocks after this sensor is deployed onto an FPGA board. This post-deployment calibration enhances LeakyDSP's robustness to different FPGA placements.

To demonstrate the viability of LeakyDSP, we first characterize the sensitivity of LeakyDSP in two experiments. In each experiment, the TDC [11] (i.e., the most studied circuits to sense voltage fluctuations) is re-implemented as a baseline and compared against LeakyDSP regarding their sensitivity. Specifically, we put LeakyDSP (resp., TDC) under a given placement and measure its sensitivity to voltage fluctuations caused by different victim activities. *Further*, considering that PDN influences its circuits unfairly, we evaluate the impact of spatial proximity to a given victim circuit on LeakyDSP (resp., TDC) by placing it onto different FPGA locations. Both experimental results show that LeakyDSP achieves fine-grained sensitivity to voltage fluctuations and strong robustness to different placements.

We then demonstrate the security implications of LeakyDSP by mounting two end-to-end case studies. First, we perform a correlation power analysis (CPA) against an AES-128 module running on FPGA, with 25 k-78 k traces for extracting the full AES key under 8 different placements of LeakyDSP and 4 operating frequencies of the AES circuit. Second, we use LeakyDSP to establish a covert channel with a high transmission rate of 247.94 b/s and low error rate of 0.24%.

Summary of contributions: The main contributions of this paper are as follows:

• We propose LeakyDSP, a new on-chip sensor design on multi-tenant FPGAs. To the best of our knowledge, we are the first to identify the information leakage through DSP blocks.

• We perform a comprehensive evaluation of LeakyDSP's capabilities. Experimental results show that LeakyDSP achieves high sensitivity to voltage fluctuations and strong robustness to different placements.

• We demonstrate the security implications of LeakyDSP, including extracting full AES keys and building covert channels.

The source code to reproduce the experiments is released at https://github.com/jjzou2002/LeakyDSP.

II. Background and Related Work

A. Power Side Channels on Multi-tenant FPGAs

FPGAs offer significant advantages for customized computation due to their inherent re-programmability and high-performance capabilities [26]. In recent years, the emergence of multi-tenant cloud FPGA environments, attracting prominent interest from both industry and academia [7], promotes flexible and cost-effective access to advanced FPGA functionalities. Specifically, a single FPGA instance is shared among numerous cloud tenants simultaneously, maximizing resource usage efficiency and potentially escalating performance. However, the sharing of an FPGA by multi-tenants, unfortunately, opens the door for various types of side channel leakages, particularly power side channels [14], [15], [29], [32], [33], [43]. Various security implications have been demonstrated via power side channels, including extracting cryptographic keys [15], [33], [43], building covert channels [9], [10], [43], fingerprinting victim computations [14], and stealing DNN model architectures [42] or inputs [25].

One key challenge in these attacks is to craft an on-chip sensor whose output is affected by its own supply voltage and to co-locate it with the victim circuit. In a multi-tenant FPGA scenario, this sensor can sense the voltage fluctuations caused by a victim circuit sharing the same PDN. All existing on-chip sensors exploit the malicious connection of traditional FPGA resources (e.g., LUTs, carry chains, wires, and FFs). As the most studied one, TDC-based circuits [11], [14] use the FPGA carry chain as a delay unit to construct a delay line, with each output connected to an FF that samples the signal traversing through the delay line. Besides, RO [42], [43], VITI [33], PPWM [32], and 1LUTSensor [15] initialize LUT and FF in their own ways. Last, RDS [29] freely places several FFs to abuse the delay incurred by routing resources, e.g., wires.

B. Digital Signal Processing Blocks

As FPGAs have transitioned from prototyping systems to deployable systems (e.g., BrainWave [23] and Catapult [24]), manufacturers have integrated digital signal processing (DSP) blocks into FPGA boards. Unlike traditional FPGA logic resources, which consume substantial area and power for complex mathematical calculations, DSP blocks offer a specialized hardware solution that significantly enhances both performance and efficiency, though at the cost of a degree of re-programmability. Accessed through dedicated hardware primitives, these DSP blocks can be configured for a wide range of arithmetic and logic operations [27].

For instance, starting with the Virtex-6 and 7-series architectures, the `DSP48E1` primitives are dedicated to the efficient processing of high-bitwidth data, enabling complex functions such as multipliers and filters to be implemented without

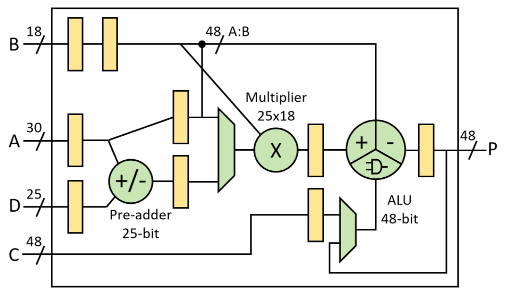


Fig. 1: Structure of the Xilinx `DSP48E1` primitive [27].

relying on traditional FPGA resources. Figure 1 shows a simplified representation of the `DSP48E1` primitive. It has four inputs with different lengths and is composed of three main sub-components, i.e., a pre-adder, a multiplier, and an arithmetic logic unit (ALU). The pre-adder is a 25-bit addition/subtraction unit that operates on input D and the lower 25 bits of input A. The multiplier multiplies the lower 25 bits of input A or the result of pre-adder with the 18-bit input B. The ALU performs addition, subtraction, and logical operations using the output from the multiplier, input C, concatenated inputs A and B, or the previous output of the DSP block. These sub-components can be flexibly configured to allow the DSP block to perform a range of functions.

III. LeakyDSP

A. Threat Model

We focus on a multi-tenant FPGA scenario where different tenants' circuits are co-located on the same FPGA. Aligned with previous FPGA power side channels [11], [14], [15], [29], [32], [33], our only assumption is that the FPGA tenant applications effectively utilize the shared PDN to optimize resource allocation and bolster overall operational efficiency. To ensure robust logical isolation, each tenant is allocated physically separate FPGA regions to deploy their circuits. The cloud service providers empower tenants with the flexibility to implement a diverse array of FPGA circuits, except those containing combinational loops [28]. Additionally, tenants can define precise placement and routing constraints tailored to the unique requirements of their own designs.

B. LeakyDSP Design

The main idea behind LeakyDSP to sense on-chip voltage fluctuations is to design a malicious DSP function that inherently responds to variations in the supply voltage. Our key observation is that the dedicated DSP blocks in FPGAs are composed of three main sub-components: a pre-adder, a multiplier, and an ALU (discussed in Section II-B). The signal propagation delay through these components is sensitive to voltage changes, leading to signal delay variations between their input and output under different voltage levels.

We note that despite the limitations in configurability compared to traditional FPGA logic resources, DSP blocks still offer sufficient flexibility to construct a voltage-sensitive function. Specifically, each of the inputs for these sub-components can be responded asynchronously without intermediate registers (i.e., flip-flops). If we connect the three components in series, their delay will be accumulated and their computation results will also have dependencies. Further, the DSP architecture supports the cascading of multiple DSP blocks. This capability allows LeakyDSP to be composed of an arbitrary number of DSP blocks, much like how a single TDC instance can utilize an arbitrary number of CARRY structures.

Figure 2 shows the overview of the proposed LeakyDSP sensor. Specifically, LeakyDSP contains two parts. First, we carefully configure the DSP blocks to make the signal propagate through the sub-components that are marked as green.

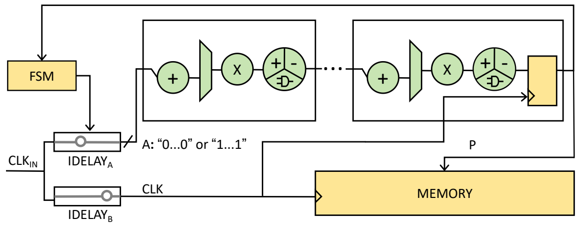


Fig. 2: An overview of LeakyDSP.

The input of the first block (A) consists of multiple bits, and each of the bits equals the delayed CLK_{IN}. Second, we leverage `IDELAY` primitive to introduce a calibration design and dynamically adjust the phase difference between input signal A and the clock signal of the flip-flop within the last DSP block CLK. By doing so, LeakyDSP's output is sensitive to voltage fluctuations and adaptive to different placements.

Malicious DSP function: In this paper, we pick the simplest function that can be instantiated by DSP blocks to analyse the information leakage of LeakyDSP, i.e., $P = A$, where A is varying between '00...0' and '11...1'. To achieve this, we first configure the pre-adder to add a constant '0' to the input (A), and pass the result $(A + 0)$ to the multiplier for further computation. Second, we set the other input of the multiplier to '1', ensuring the multiplication result $(A + 0) \times 1$ remains consistent with the original input A. Third, we configure the ALU as an adder and set the second input as a constant '0', producing a result of $(A + 0) \times 1 + 0$. Last, we configure the output mode based on the position of the DSP block. If it is the last DSP block in a LeakyDSP instance, the output of ALU passes through a register before being stored in memory. Otherwise, the lower bits of output are fed directly into the next DSP block as its input.

Extracting the intensity of the voltage fluctuations: When the PDN is idle, LeakyDSP's outputs remain stable with only a few bit flips. However, in the case of a busy PDN, the number of flipped bits increases in response to fluctuating voltage. Taking `DSP48E1` as an example, its output is 48-bit wide, enabling a fine-grained quantization to voltage levels. To this end, LeakyDSP converts each raw sampled value into a numerical representation based on its Hamming weight [11], [29], referred to as the *readout*.

Calibration: Aligned with existing crafted circuits [11], [15], [29], [32], [33], LeakyDSP contains a calibration part to ensure that the bit-flip number from LeakyDSP's outputs changes accordingly when the voltage in PDN starts fluctuating. To dynamically adjust the initial delay, LeakyDSP incorporates two `IDELAY` components, which are part of the SelectIO resources and designed for signal timing adjustment in FPGA devices. Each `IDELAY` can generate a runtime-adjustable delay of up to the half of the period of its connected clock signal (i.e., $T/2$) on the signal passing through it. By connecting them to the input signal A and clock signal CLK respectively, LeakyDSP can achieve a phase difference ranging from $-T/2$ to $T/2$ between A and CLK. We adopt a straightforward calibration procedure for LeakyDSP. Specifically, we iteratively increase the initial clock delay by changing the configuration of the two `IDELAY` units, until the readouts of LeakyDSP reach the maximum variation between two consecutive adjustments.

Implementation: Though LeakyDSP can be instantiated with an arbitrary number of used DSP blocks (i.e., n), the optimal choice of n is influenced by the resource budget and sensitivity to voltage fluctuations. In this paper, we select $n = 3$, an empirical value that provides a balance of high sensitivity, acceptable resource usage, and ease of calibration. Besides, we use the `DSP48E1` primitive to map the configuration of DSP blocks on 7-series FPGA boards and `DSP48E2` on Zynq Ultrascale+ FPGAs. For delay configurations, we employ `IDELAYE2` on 7-series FPGAs and `IDELAYE3` on Zynq UltraScale+ FPGAs.

IV. Evaluation

In this section, we characterize LeakyDSP and demonstrate how LeakyDSP can be used to mount end-to-end attacks.

Machine settings: To perform the characterization and AES key extraction attack, we use a Basys3 board connected to a Huawei MateBook 14 laptop via UART and powered through the USB interface. To evaluate the covert channel, we use an ALINX AXU3EGB board, which has the similar architecture as the AWS EC2 instance (i.e., Xilinx UltraScale+ VU9P FPGA) [3]. To deploy hardware code onto the FPGA platforms, we use Vivado version 2020.1 for synthesis, implementation, and bitstream generation. Unless otherwise stated, all experiments are conducted using the default Vivado synthesis and implementation settings.

A. Characterizing LeakyDSP

To show the sensitivity of LeakyDSP to voltage fluctuations, we opt for power virus circuits as a victim to generate varying voltage fluctuations, aligned with [43]. The power virus circuit consists of a number of instances that are implemented as RO circuits. Specifically, within the power virus circuits, each RO circuit consists of a single inverter, an AND gate, and an FF. The output of the inverter is controlled by configuring the enable signal, determining whether it is active to make the output signal frequently switch between '0' and '1'. These switching activities can cause a voltage fluctuation in PDN.

LeakyDSP's sensitivity under different victim activities: Our first experiment is to characterize the readout change of LeakyDSP to various victim activities. Specifically, we create 8,000 instances of power virus to cover over 33.3% routing places of our Basys3 board and activate about 46% of available LUT resources. We then divide them into 8 groups and each group has 1000 evenly-distributed instances. When we activate a varying number of these groups, the voltage varies from 9 levels. For each level, we collect 2,000 readouts from LeakyDSP and take the average of them as the final result to compute Pearson correlation coefficient and regression coefficients. Pearson correlation coefficient is used to assess the strength and direction of the linear association, while the regression coefficient measures the impact of these two values in a linear regression model.

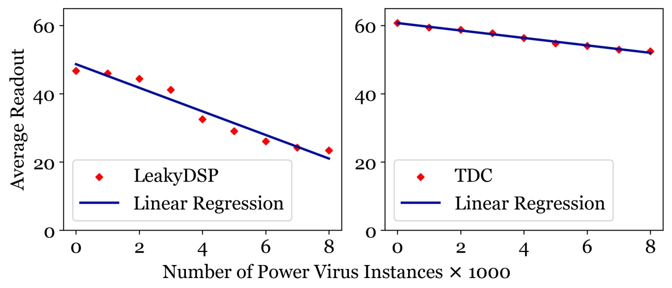
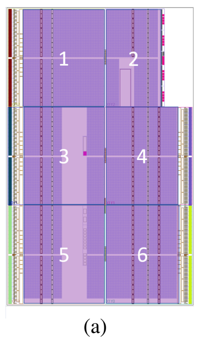


Fig. 3: The sensitivity of LeakyDSP and TDC in the same placement under different victim activities. Note that we implement TDC with 128 FFs, resulting in a 128-bit output, while LeakyDSP produces a 48-bit output.



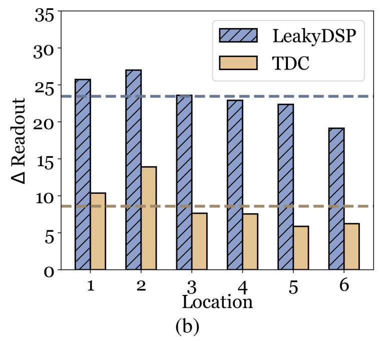


Fig. 4: The sensitivity of LeakyDSP and TDC [11] under different placements. The index as the x-axis in Figure 4(b) corresponds to the region index in Figure 4(a). The dashed line indicates the average value of a sensor.

Figure 3 shows the relationship between these readouts (i.e., the number of unflipped bits in LeakyDSP's output) and the number of activated power virus instances. These two values of LeakyDSP exhibit a roughly linear relationship with a Pearson correlation coefficient of -0.974, while TDC achieves -0.996. The close proximity of these correlation coefficients to -1 indicates a strong linear relationship, demonstrating that all the evaluated voltage levels fall within the observable range of LeakyDSP and can be accurately estimated using its readout. Furthermore, compared with the most studied on-chip sensor (i.e., TDC), LeakyDSP achieves a high regression coefficient of -3.45, while TDC achieves -1.09. This indicates that LeakyDSP provides finer granularity under the given sensor placement.

LeakyDSP's sensitivity under different placements: In a multi-tenant FPGA, spatial proximity between a crafted circuit and a victim's circuit is beyond the attacker's control. As such, we characterize the readout change of LeakyDSP with respect to their spatial proximity to victim logic that consumes power. Specifically, we instantiate 8,000 instances of the power virus, and constrain the placement of the power virus to region 1 and region 2 in Figure 4(a). Subsequently, we put LeakyDSP (resp., TDC) in each of the 6 FPGA regions via a `Pblock` constraint with all the 8,000 power virus instances off and on.

We sample 2,000 readouts under each setting and calculate their average.

As shown in Figure 4, although the effectiveness of LeakyDSP is influenced by the unevenness of the FPGA's power supply design, it can still sense the given voltage fluctuation under all the above placements. Specifically, LeakyDSP achieves the best performance when it is placed in region 2. Further, when LeakyDSP is placed under the worst-case placements (i.e., region 5 and region 6 which are far away from the victim circuit), its readouts are still sensitive to voltage fluctuations.

B. Stealing Cryptographic Keys

As the most widely adopted symmetric-key cryptographic algorithm, AES has been implemented in different devices, including hardware circuits on FPGA platforms [1], [20], [38]. However, prior work has demonstrated that a remote attacker can extract the full secret key of AES circuits via CPA attacks [8]. These attacks leverage the Pearson Correlation Coefficient [6] to detect linear dependencies between voltage fluctuations during AES circuit execution and hypothetical power calculated from the ciphertext with a guessed key. To achieve this, attackers first use the Hamming weight and Hamming distance power models to estimate the power consumption for each possible key value. Then, they use their own side channel to obtain the power-related information. Last, they sort each guessed value according to its correlation coefficient to determine the ground-truth key value.

Experimental setup: Our end-to-end attack is built upon the open-source code of RDS [29], which implements an AES-128 module [1] that co-resides with the attack circuit on the Basys3 board. By default, the clock frequencies of the attack circuit and the AES are set to 300 MHz and 20 MHz, respectively. Please note that 20 MHz is exactly aligned with the victim setting in VITI [33] and RDS [29], and we just aim to show the general possibility of leakage. Unless otherwise stated, we collect 60 k power side-channel traces for each setting. In each trace collection, we transmit the key `K` and the plaintext `PT` to the AES core, and record the LeakyDSP's readouts obtained during AES encryption. To avoid plaintext repetition, we employ the current ciphertext as the subsequent plaintext. To simplify the process, we utilize the start encryption signal to trigger the collection of a sensor trace. To evaluate the performance of LeakyDSP, we run the key extraction attack under 8 placements and 4 different AES clock frequencies.

Aligned with previous AES key recovery attacks [15], [29], [32], [33], we use the key rank (KR) estimation metric, which evaluates the number of key candidates that an attacker needs to test. Specifically, the key rank is presented as a narrow range with an upper and lower bound, due to the bounded error in current key rank estimation algorithms. For instance, in the absence of any side-channel information, the upper key rank corresponds to the entire key space, which is 2^{128} for AES-128. Conversely, when the full key is compromised, the lower bound of the KR reduces to one, while the upper bound tightly constrains the range, indicating that the correct key can be deduced with minimal guessing. In such scenarios, attackers can arrange each guessed byte value based on correlation coefficient to obtain the genuine AES key. To speed up computation, we employed the open-source analysis tool [8] for conducting our CPA analysis on an NVIDIA GeForce RTX 3090 GPU.

TABLE I: Number of traces required to break the full AES 128-bit key for different sensor placements.

Setting	**Traces to Extract Full Key** ($\times 1000$)									
	P1	P2	P3	P4	P5	P6	P7	P8	Average	Baseline
LeakyDSP	55	43	58	28	48	25	37	41	41	51

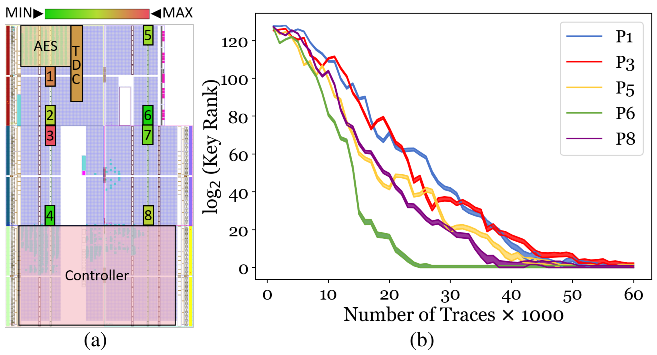


Fig. 5: Key rank estimation for LeakyDSP. The curves in Figure 5(b) are named after the number on each placement shown in Figure 5(a).

Experimental result: Figure 5 shows the average upper and lower bounds of the key rank estimation metric to evaluate the impact of the placement of LeakyDSP. Figure 5(a) is a color gradient that rates all 8 placements based on their key rank with 20 k power traces, revealing that the performance of a power analysis attack relies on the placement of LeakyDSP. This observation aligns with the findings reported in [29], [33], where the location-dependent sensitivity is attributed to the non-uniformity of the PDN across the FPGA board. Moreover, Figure 5(b) shows the key rank variability of LeakyDSP across 5 selected placements out of the total 8 placements, including the best and worst-case placements, as well as the placement closest to the victim circuit. Each placement is identified by its corresponding number, as indicated adjacent to them in Figure 5(a). The full results are summarized in Table I. Compared to TDC, which requires 51 k traces to extract the full key, LeakyDSP achieves a comparable efficiency with 25 k-58 k traces needed. We do not claim that our LeakyDSP is better than TDC in terms of the attack performance, as we only evaluate TDC in one setting. As our LeakyDSP does not use any traditional logic resources inside configurable logic blocks, we cannot place LeakyDSP and TDC in the same position for a solid comparison [29], [33].

We then evaluate the impact of the AES frequency on our attack, utilizing the best-case placement for attackers (i.e., P6). Figure 6 illustrates the corresponding key rank outcomes.

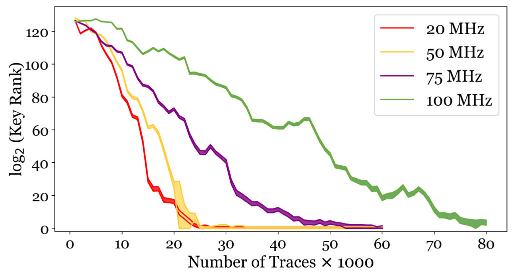


Fig. 6: The impact of the AES frequency on our attack. The efficiency of key extraction decreases as the frequency of the victim circuit increases.

We note that the performance of the attack is dependent on the frequency. The efficiency of our key extraction decreases as the frequency of the victim circuit increases. At a high frequency of 100 MHz, the 128-bit AES key cannot be fully recovered even with all 60 k traces. To address this, we collect an additional 20 k traces, achieving full key recovery with a total of 78 k traces.

C. Establishing Covert Channels

Aligned with existing work [10], [13], we apply the covert channel for assessing the bandwidth of our side channels, where the attacker has complete control over both the sender and receiver. We assume the colluding sender and receiver operate on the same FPGA. To induce the voltage fluctuations for the sender, we opt for power virus circuits as to generate varying voltage fluctuations. To transmit a ‘1’, the sender sets its enable signals to ‘0’, making all the power virus instances idle. To transmit a ‘0’, a sender activates all the controlled power virus instances to plunder the voltage of receiver circuits. The receiver is a LeakyDSP-based circuit and loops to monitor voltage level via its readouts. By comparing the averaged readouts to a predefined threshold, the receiver distinguishes between the transmitted bits, ‘0’ or ‘1’.

Experimental setup: To assess the performance of our covert channel, we vary the time for sending one bit from 2 ms to 7.5 ms and send 10 kb of random data in each configuration. Besides, we employ two metrics that are widely used in previous covert channels [41]: transmission rate (TR) and bit error rate (BER). TR measures the amount of data transmitted per unit time, while BER represents the proportion of received bits incorrectly interpreted relative to the sent bits.

Experimental result: Figure 7 shows the end-to-end results across 10 runs. The experimental results show a negative correlation between the BER and the time for sending a bit. When the sending time is greater than 3.5 ms, the BER stabilizes below 1%. When the sending time is less than 3 ms, the BER gradually increases with the decreasing of sending time due to the limited accuracy of the receiver in distinguishing data. The smaller the time for sending a bit, the greater the TR. So we recommend the sending time of 4 ms with a BER of 0.24%. With such setting, the TR of a single transmission is 247.94 b/s.

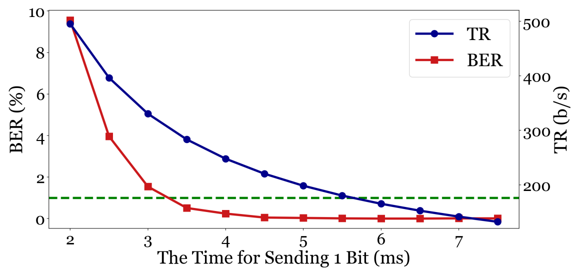


Fig. 7: The performance impact of timing parameters on LeakyDSP-based covert channel.

V. Discussion

Mitigation: Similar to existing on-chip sensors, LeakyDSP relies on precise voltage information from a shared PDN. While enhancing the PDN could mitigate such attacks, this approach is limited by current production technologies [30], [35] and does not protect already-deployed products. An alternative countermeasure is to inject noise into the PDN [12], [17], which obscures power patterns but introduces performance overhead. Besides, the developers can modify their circuits as constant-power implementation with some efforts. Last, after identifying the threats from DSP blocks, cloud providers can prevent the deployment of LeakyDSP by enforcing synchronized inputs or mandatory timing checks on DSP configurations. However, we note that both approaches limit benign tenants’ flexibility in DSP reconfiguration. Besides, timing violation checks can be bypassed using programmable clock-generating circuits, a method applicable to TDC [11] and RDS [29] that face the same timing violation issues.

Future work: This paper demonstrates that DSP blocks can indeed be exploited for power analysis attacks through a comprehensive evaluation on two FPGA boards with different architectures. However, while LeakyDSP’s readouts vary with voltage levels monotonously, this change is not absolutely uniform. Due to the *black-box* nature of DSP blocks’ internal design, we cannot entirely explain LeakyDSP’s leakage behavior at this stage. Besides, the number of DSPs within a single LeakyDSP sensor is an empirical value ($n = 3$). We leave the exploration of DSP operating principles and the optimal selection of n as future work.

VI. Conclusion and Acknowledgments

In this paper, we propose LeakyDSP, a novel on-chip sensor that maliciously configures DSP blocks to sense voltage fluctuations on multi-tenant FPGAs. Our characterization shows that LeakyDSP is highly sensitive to voltage fluctuations and robust across different placements. The viability of LeakyDSP is demonstrated by two case studies, which extract full AES keys under various placements of LeakyDSP and operating frequencies of victim circuits, and build FPGA-to-FPGA covert channels with a high transmission rate of 247.94 b/s.

We thank the anonymous reviewers for their feedback. In addition, we thank Yuhui Zhang for her valuable comments and discussions that helped us to improve the work. This work was supported by XXX.

References

[1] AIST and T. University, "Aes encryption core," 2007. [Online]. Available: http://www.aoki.ecei.tohoku.ac.jp/crypto/

[2] Alibaba. (2024) Compute optimized instance families with fpgas. [Online]. Available: https://www.alibabacloud.com/help/en/fpga-as-a-service/product-overview/fpga-accelerated-compute-optimized-instance-families

[3] Amazon, "Amazon ec2 f1 instances," 2024. [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1

[4] A. AWS., "Amazon ec2 f1," 2024. [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/

[5] M. Azure. (2024) Machine learning. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/machine-learning/

[6] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

[7] G. Dessouky, A.-R. Sadeghi, and S. Zeitouni, "Sok: Secure fpga multi-tenancy in the cloud: Challenges and opportunities," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021.

[8] H. Gamaarachchi, H. Ganegoda, and R. Ragel, "The a to z of building a testbed for power analysis attacks," in *International Conference on Industrial and Information Systems*, 2015.

[9] I. Giechaskiel, K. Rasmussen, and J. Szefer, "Reading between the dies: Cross-slr covert channels on multi-tenant cloud fpgas," in *IEEE 37th International Conference on Computer Design (ICCD)*, 2019, pp. 1–10.

[10] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "C3apsule: Cross-fpga covert-channel attacks through power supply unit leakage," in *IEEE Symposium on Security and Privacy*, 2020, pp. 1728–1741.

[11] O. Glamočanin, L. Coulon, F. Regazzoni, and M. Stojilović, "Are cloud fpgas really vulnerable to power analysis attacks?" in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 1007–1010.

[12] O. Glamočanin, A. Kostić, S. Kostić, and M. Stojilović, "Active wire fences for multitenant fpgas," in *International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2023, pp. 13–20.

[13] D. R. E. Gnad, C. D. K. Nguyen, S. H. Gillani, and M. B. Tahoori, "Voltage-based covert channels using fpgas," *ACM Trans. Des. Autom. Electron. Syst.*, 2021.

[14] M. Gobulukoglu, C. Drewes, W. Hunter, R. Kastner, and D. Richmond, "Classifying computations on multi-tenant fpgas," in *Design Automation Conference*, 2021, pp. 1261–1266.

[15] D. Jayasinghe, B. Udugama, and S. Parameswaran, "1lutsensor: Detecting fpga voltage fluctuations using lookup tables," *Cryptographic Hardware and Embedded Systems*, pp. 51–86, 2024.

[16] D. Korolija, T. Roscoe, and G. Alonso, "Do OS abstractions make sense on FPGAs?" in *USENIX Symposium on Operating Systems Design and Implementation*, 2020, pp. 991–1010.

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

[18] J. Krautter, D. R. Gnad, and M. B. Tahoori, "Mitigating electrical-level attacks towards secure multi-tenant fpgas in the cloud," *ACM Transactions on Reconfigurable Technology and Systems*, 2019.

[19] J. Landgraf, T. Yang, W. Lin, C. J. Rossbach, and E. Schkufza, "Compiler-driven fpga virtualization with synergy," in *Architectural Support for Programming Languages and Operating Systems*, 2021.

[20] R. Lumbiarres-López, M. López-García, and E. Cantó-Navarro, "Hardware architecture implemented on fpga for protecting cryptographic keys against side-channel attacks," *IEEE Transactions on Dependable and Secure Computing*, pp. 898–905, 2018.

[21] J. Ma, G. Zuo, K. Loughlin, X. Cheng, Y. Liu, A. M. Eneyew, Z. Qi, and B. Kasikci, "A hypervisor for shared-memory fpga platforms," in *Architectural Support for Programming Languages and Operating Systems*, 2020, p. 827–844.

[22] J. M. Mbongue, D. T. Kwadjo, A. Shuping, and C. Bobda, "Deploying multi-tenant fpgas within linux-based cloud infrastructure," *ACM Trans. Reconfigurable Technol. Syst.*, 2021.

[23] Microsoft. (2024) Project brainwave. [Online]. Available: https://www.microsoft.com/en-us/research/project/project-brainwave/pricing/details/machine-learning/

[24] ——. (2024) Project brainwave. [Online]. Available: https://www.microsoft.com/en-us/research/project/project-catapult/

[25] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Remote power side-channel attacks on bnn accelerators in fpgas," in *Design, Automation & Test in Europe Conference & Exhibition*, 2021.

[26] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, and Y. Zhou, "Mlperf inference benchmark," in *International Symposium on Computer Architecture*, 2020, pp. 446–459.

[27] B. Ronak and S. A. Fahmy, "Mapping for maximum performance on fpga dsp blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[28] A. W. Services. (2024) Aws ec2 fpga hdk+sdk errata. [Online]. Available: https://github.com/aws/aws-fpga/blob/master/ERRATA.md

[29] D. Spielmann, O. Glamočanin, and M. Stojilović, "Rds: Fpga routing delay sensors for effective remote power analysis attacks," *Cryptographic Hardware and Embedded Systems*, 2023.

[30] B.-H. Su, J.-S. Tang, H.-J. Lee, and C.-B. Tzeng, "Noise analysis and improvement of power supply network based on power integrity," in *International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT)*, 2023, pp. 317–320.

[31] T. Sugawara, K. Sakiyama, S. Nashimoto, D. Suzuki, and T. Nagatsuka, "Oscillator without a combinatorial loop and its threat to fpga in data centre," *Electronics Letters*, vol. 55, no. 11, pp. 640–642, 2019.

[32] B. Udugama, D. Jayasinghe, H. Saadat, A. Ignjatovic, and S. Parameswaran, "A power to pulse width modulation sensor for remote power analysis attacks," *Cryptographic Hardware and Embedded Systems*, pp. 589–613, 2022.

[33] Udugama, Brian and Jayasinghe, Darshana and Saadat, Hassaan and Ignjatovic, Aleksandar and Parameswaran, Sri, "Viti: A tiny self-calibrating sensor for power-variation measurement in fpgas," in *Cryptographic Hardware and Embedded Systems*, 2022.

[34] Z. Wang, G. Zhu, Y. Liu, Y. Chang, K. Zhang, and M. Chen, "Xuni: Virtual machine abstraction for self-contained and multi-tenant cloud fpgas," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2024.

[35] D. Xiao, X.-j. Li, Y. Li, H. Lin, J. Qian, and C. Hu, "The screening method of board-level decoupling capacitors for multi-power distribution network," in *Chinese Control Conference*, 2023, pp. 7111–7116.

[36] Xilinx, "Vivado design suite user guide: Dynamic function exchange (ug909)," 2024. [Online]. Available: Available:https://docs.xilinx.com/r/en-US/ug909-vivado-partial-reconfiguration/Introduction-to-Dynamic-FunctioneXchange

[37] Xilinx, "Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency," 2024. [Online]. Available: https://docs.xilinx.com/v/u/en-US/wp380_Stacked_Silicon_Interconnect_Technolog

[38] J. Zambreno, D. Nguyen, and A. Choudhary, "Exploring area/delay tradeoffs in an aes fpga implementation," in *International Conference on Field Programmable Logic and Applications*, 2004, pp. 575–585.

[39] Y. Zha and J. Li, "Virtualizing fpgas in the cloud," in *Architectural Support for Programming Languages and Operating Systems*, 2020.

[40] F. Zhang, Z. Wang, H. Shen, B. Yang, Q. Wu, and K. Ren, "Darpt: defense against remote physical attack based on tdc in multi-tenant scenario," in *Design Automation Conference*, 2022, p. 559–564.

[41] X. Zhang, Z. Zhang, Q. Shen, W. Wang, Y. Gao, Z. Yang, and Z. Wu, "Thermalscope: A practical interrupt side channel attack based on thermal event interrupts," in *Design Automation Conference*, 2024.

[42] Y. Zhang, R. Yasaei, H. Chen, Z. Li, and M. A. Al Faruque, "Stealing neural network structure through remote fpga side-channel analysis," *IEEE Transactions on Information Forensics and Security*, 2021.

[43] M. Zhao and G. E. Suh, "Fpga-based remote power side-channel attacks," in *IEEE Symposium on Security and Privacy*, 2018.