MUXLeak: Exploiting Multiplexers as A Power Side Channel against Multi-tenant FPGAs

Xin Zhang, Jiajun Zou, Zhi Zhang, Qingni Shen, Yansong Gao, Jinhua Cui, Yusi Feng, Zhonghai Wu, Derek Abbott, *Fellow, IEEE*

***Abstract*— FPGA cloud acceleration, or "FPGA as a Service" (FaaS), offered by AWS, Microsoft Azure, Alibaba Cloud, and Huawei Cloud, has become a promising solution for tackling complex, compute-intensive workloads. It targets applications such as genomics, image and video processing, electronic design automation, compression, and big data analytics. While multi-tenant FPGAs significantly enhances resource utilization efficiency, it faces security threats from power side channels, where attackers craft a malicious circuit to detect voltage fluctuations from victim circuits. Observing that all the crafted circuits exploit either Carry Chain or Look-up Table to sense voltage fluctuations, existing defenses have focused on detecting the malicious use of the two basic FPGA computing resources. However, it remains unclear whether such countermeasures are sufficient to address the growing threat of power side channels in multi-tenant FPGAs.**

In this paper, we reveal MUXLeak, a novel on-chip sensor that exploits *Multiplexer (MUX)* to craft a stealthy power side channel, which bypasses existing countermeasures. Particularly, we perform a thorough analysis of basic resources within an FPGA unit and unveil that *MUX*, another basic resource, *has never been exploited before*. More importantly, it can be directly initialized on Xilinx FPGAs and its incurred signal propagation delay demonstrates an inverse correlation with changes in voltage, making itself exploitable for a new power side channel leakage. In our evaluation, we test MUXLeak on three Xilinx FPGA products and use TDC [18] (i.e., the most sensitive on-chip sensor until now) to benchmark the sensitivity of MUXLeak. Our results show that MUXLeak has achieved the same level of sensitivity as TDC to voltage fluctuations. Further, we apply MUXLeak to mount two attacks, i.e., extracting AES keys within 2.54 hours and stealing DNN model architectures with an accuracy of over 90%.

Keywords—Multi-tenant FPGA, Cloud Computing, Side Channel, Remote Power Analysis

Corresponding authors: Zhi Zhang and Qingni Shen.

Xin Zhang, Jiajun Zou, Qingni Shen, and Zhonghai Wu are with the School of Software and Microelectronics, Peking University, Beijing 100871, China, also with the National Engineering Research Center for Software Engineering, Peking University, Beijing 100871, China, and also with the PKU-OCTA Laboratory for Blockchain and Privacy Computing, Peking University, Beijing 100871, China. E-mail: qingnishen@ss.pku.edu.cn.

Zhi Zhang and Yansong Gao are with the Department of Computer Science and Software Engineering, The University of Western Australia, Perth, WA 6009, Australia. E-mail: zzhangphd@gmail.com.

Jinhua Cui is with the College of Semiconductors (College of Integrated Circuits), Hunan University, Changsha 410082, China.

Yusi Feng is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, 518055, China.

Derek Abbott is with the School of Electrical and Electronic Engineering, The University of Adelaide, Adelaide, SA 5005, Australia.

I. Introduction

In recent years, major commercial cloud service providers such as Amazon AWS, Alibaba, Huawei, and Microsoft Azure have provided remote access to high-performance Field-Programmable Gate Arrays (FPGAs), known as FPGA-as-a-service (FaaS) [9]. Different from CPUs and GPUs which optimize instruction streams for acceleration, FPGAs allow users to directly reconfigure the underlying gate-level hardware resources, thereby achieving notable improvements in performance and energy efficiency [39], [35], [53]. For example, FlightVGM [32], a state-of-the-art FPGA accelerator for video generation model (VGM) inference, demonstrates these advantages. When deployed on the AMD V80 FPGA, it achieves $1.30\times$ higher performance and $4.49\times$ greater energy efficiency on sparse VGM workloads compared to that of the NVIDIA RTX 3090 GPU.

To further optimize 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 [10]. Particularly, multiple end-to-end frameworks have been proposed to virtualize FPGA resources on the cloud, including Coyote [27], OPTIMUS [34], SYNERGY [31], and XUNI [49]. Besides, industrial advancements from Xilinx such as Stacked Silicon Interconnect (SSI) and Dynamic Function eXchange (DFX) have been integrated into high-end FPGAs such as Xilinx Alveo U280 to facilitate multi-tenant FPGAs.

Given that FPGA multi-tenancy has attracted considerable attention from both academia and industry, this threat model faces new security concerns raised by recent works [58], [38], [10], [33], among which, side channel attacks are the mainstream concern. Particularly, in a multi-tenant FPGA platform, an attacker can co-reside with a targeted victim and share the same power delivery network (PDN) [9]. As current PDNs are unable to eliminate variations in current when supplying power to a circuit [59], this results in varying transient voltage drops, causing delays in signal propagation.

To this end, the attacker can craft a circuit[1] to sense the delay variations caused by voltage fluctuations from the victim circuit, posing various power side-channel threats [58], [40], [51], [57], [37], [15], such as extracting cryptographic keys [58], [40], [15], and recovering DNN model architectures/inputs [51], [57], [37]. All these attacks are realistic in that they eliminate the need for knowing the targeted victim's circuit design and location in FPGA, aligned with

[1]We use "crafted circuit" and "on-chip sensor" interchangeably.

the aforementioned threat model where tenants do not know each other in a multi-tenant FPGA platform. Since each of these attacks exploit a basic FPGA computing resource, they can be categorized as *Carry Chain* (CARRY) and *Look-up Table* (LUT)-based crafted circuits. CARRY facilitates fast carry propagation and LUT enables the implementation of diverse logic functions. The signal delay incurred by either structure is sensitive to voltage.

To mitigate the aforementioned attacks, a number of defenses have been proposed [36], [44], [3], [30], [29] in detecting malicious use of the CARRY or LUT structures. First, the CARRY-based crafted circuits [60], [40], [37], [51] require a uniquely long chain of basic CARRY structures, making them easily identifiable by Design Rule Check [44], [15]. Further, these circuits incur a significant area overhead and are limited in vertical placement, posing challenges for their deployment on multi-tenant FPGAs [36]. Second, the LUT-based crafted circuits [58], [57], [13] (so-called *RO* circuits) present a distinctive characteristic of *combinational loop*, rendering themselves detectable [3], [30], [29], [41]. For example, Amazon AWS has banned these RO circuits by leveraging this characteristic [41]. To this end, we ask the following research questions:

Is there any other common FPGA resource that can be exploited to craft a *new* and *stealthy* side channel?

In this paper, we provide positive answers by presenting MUXLeak, a new on-chip sensor that exploits *Multiplexers* (MUXes) to craft a stealthy power side channel on multi-tenant FPGAs, bypassing all existing countermeasures. It can be used to collect fine-grained voltage information from a targeted victim, without details about the victim's circuit design or location in the FPGA board.

We observe that both the CARRY and LUT structures from existing on-chip sensors are basic FPGA computing resources, each of which consists of CMOS transistors. What they have in common is that both are components of a *Configurable Logic Block (CLB)*, an FPGA unit for allocating resources to a circuit. With this observation, we have performed a thorough analysis of all components within a CLB of Xilinx FPGAs. Particularly, a CLB has 4 components including LUT, CARRY, *Flip-Flop (FF)* and *Multiplexer (MUX)*. While FF contains complementary MOSFETs [24], [25] and is used by existing crafted circuits to store information, MUX is composed of CMOS transistors and *has never been explored before at the CLB level*. Further, its incurred signal propagation delay demonstrates an inverse correlation with changes in voltage, rendering itself potentially exploitable for a new side channel.

To this end, we design a new on-chip sensor, called MUXLeak, which consists of MUXes without using any CARRY and LUT structures that are in the spotlight of existing defenses. When the voltage fluctuates, the signal propagation delay from MUXes changes accordingly, resulting in different numbers of bit flips in the output stream of MUXLeak. Specifically, MUXLeak has groups of chained MUXes with each group connecting its MUXes via a different routing path. By doing so, MUXLeak generates a fine-grained scale of delay, rendering itself sensitive to changes in voltage. As the change in delay decides the bit-flip number in the output, MUXLeak uses the bit-flip number to sense fluctuations in voltage. To ensure and optimize MUXLeak's sensitivity across its different placements on various FPGA boards, we further propose a calibration design that carefully calibrates the initial delay of MUXes without a victim circuit involved. Our design leverages MUXes to enable a dynamic configuration of routing paths. This indicates that the calibration can be decided after MUXLeak is deployed onto an FPGA board, making MUXLeak sensitive in different placements.

To demonstrate the viability of MUXLeak, we evaluate it on three Xilinx FPGA boards (i.e., PYNQ-Z2, ALINX AXU3EGB, and Basys 3)[2]. We first characterize the sensitivity of MUXLeak in two experiments. In each experiment, TDC [18] (i.e., the most sensitive on-chip sensor until now) is re-implemented as a benchmark and compared against MUXLeak regarding their sensitivity. Specifically, we put MUXLeak (resp., TDC) under a given placement and measure its sensitivity to voltage fluctuations caused by different victim activities (i.e., different numbers of activated power virus instances ranging from 0 to 16,000). *Further*, considering that PDN influences its circuits unfairly, we evaluate the impact of spatial proximity to a given victim circuit (i.e., 8000 activated power virus instances) on MUXLeak (resp., TDC) by placing it onto different FPGA locations. Both experimental results show that MUXLeak has achieved the same level of sensitivity as TDC [18] to voltage fluctuations.

We then demonstrate the security implications of MUXLeak by mounting different attacks. *First*, we demonstrate a correlation power analysis (CPA) against an AES-128 module, within 2.54 hours for extracting the full AES key under 8 different placements of MUXLeak and 4 operating frequencies of the AES circuit. *Second*, we apply MUXLeak to steal DNN model architectures with an accuracy of over 90%.

Contribution: The main contributions of this paper are summarized as follows:

- After a thorough analysis of basic FPGA resources, we have found a widely used but never exploited structure (i.e., MUX), that can be used to craft a new and stealthy on-chip sensor, coined as MUXLeak.
- We perform a comprehensive evaluation of MUXLeak's capabilities. Experimental results show that MUXLeak achieves the same level of sensitivity as the most sensitive on-chip sensor (i.e., TDC [18]) to voltage fluctuations.
- We demonstrate the security implications of MUXLeak using 2 case studies, including extracting full AES keys and stealing DNN model architectures.

The source code to reproduce our experiments is released at https://github.com/zhangxin00/MUXLeak-artifact/.

II. Background and Related Work

A. Basic FPGA Resources

In Xilinx FPGAs, the configurable logic block (CLB) is a basic unit used to allocate resources to a circuit [42]. The

[2]Xilinx is the largest FPGA supplier with about 45% of the market share [1]

CLB is connected to a switch matrix for access to the general routing, providing a configurable and programmable fabric within FPGA. Once the logic circuit design is finalized using hardware description languages, Electronic Design Automation (EDA) tools (e.g., Vivado and Quartus) assign CLBs within the FPGA. These assignments directly correspond to the circuit designs, meticulously transforming the digital blueprint into implemented logic circuits.

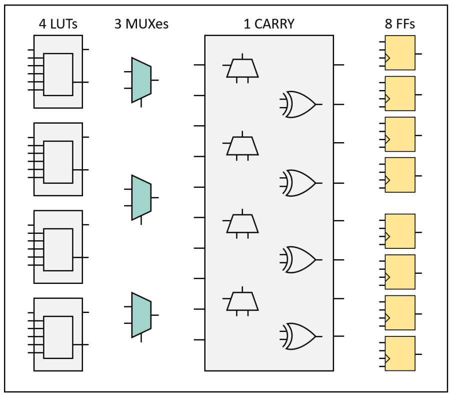


Fig. 1: Each slice in the Xilinx 7 series architecture encompasses four types of basic resources: 4 LUTs, 1 CARRY, 8 FFs, and 3 MUXes. Two slices are combined to form a CLB.

To allocate resources more selectively and efficiently, a CLB is implemented as one or two slices [24], [25], depending on a specific architecture. In the Xilinx 7 series FPGA [24], a CLB consists of two slices. As shown in Figure 1, each slice contains 4 Look-up Tables (LUTs) and 8 FFs, along with a dedicated Carry Chain (CARRY) and 3 MUXes. In the UltraScale series FPGAs [25], CLBs have combined the two slices into one, which contains all 8 LUTs and 16 FFs, along with 1 CARRY and 7 MUXes.

Despite the fact that these four components internally consist of more intricate AND gates, OR gates, and multiplexers, they represent the basic resources that users can directly access. When deploying a hardware design on an FPGA board, developers are required to map their logical design to these four basic resources via EDA tools to generate a bitstream file. Additionally, only these four types of basic resources can be instantiated directly by invoking the relevant hardware primitives.

First, LUTs are used to implement various function generators, effectively replacing the role of logic gates in traditional circuit design. By initializing the LUT through exhaustively listing the output values of various input situations, the corresponding top-level functions can be realized. Second, MUXes play a crucial role in controlling the signal routing within CLBs, e.g., expanding the number of inputs in a circuit via changing signal propagation paths. Third, CARRYs are designed to improve the performance of arithmetic functions such as adders, counters, and comparators. FPGA designs that include simple counters or adders /subtractors automatically invoke this CARRY resource for enhanced speed and efficiency. Last, FFs serve as storage elements within the FPGA architecture. By employing complementary MOSFETs to establish positive feedback loops, FFs maintain their output state until a specific triggering condition is met. This characteristic endows FFs with the ability to reliably store and retain data.

B. Multiplexers on Xilinx FPGAs

As Multiplexers (MUXes) allow for efficient switching and transmission of hardware signals with few transistors [6], they are widely implemented at different levels in FPGA-based systems. At the interconnect level, they are used as routing resources to build interconnection between CLBs and other components in the FPGA fabric. At the CLB level, MUXes serve to facilitate the implementation of logic functions by developers by changing signal propagation paths. They can be expanded to support 4:1 MUXes, 8:1 MUXes, or 16:1 MUXes by combining varying numbers of LUTs. Last, although the internal details of LUT and CARRY are not transparent to developers, a specific number of multiplexers are used to construct their functional logic.

This paper focuses on the MUXes at the CLB-level, which typically number above 10,000 in an FPGA board [24], [25] and can be further categorized into various types. In the Xilinx 7 series FPGA architecture, each CLB contains 3 types of MUXes, i.e., F7AMUX, F7BMUX, and F8MUX. In the Ultrascale architecture, each CLB contains 7 types of MUXes, namely F7MUX_AB, F7MUX_CD, F7MUX_EF, F7MUX_GH, F8MUX_BOT, F8MUX_TOP, and F9MUX. Despite these MUXes have distinct both locations and input sources, all of them share a common structure, which includes two data inputs $\mathtt{I}_0$ and $\mathtt{I}_1$, one select signal $\mathtt{S}$, and one data output $\mathtt{O}$. Upon asserting $\mathtt{S}$ to 0, the MUX outputs $\mathtt{O}$ corresponds to $\mathtt{I}_0$, while asserting $\mathtt{S}$ to 1 associates the output $\mathtt{O}$ with $\mathtt{I}_1$. When mapping a circuit design onto an FPGA board, developers can utilize the EDA tools to handle the automatic assignment of the relevant MUXes. Alternatively, by invoking pertinent hardware primitives, developers can directly instantiate a specific type of MUXes.

C. Power Side Channels on Multi-tenant FPGAs

In recent years, FPGAs offer significant advantages for customized computing due to their inherent re-programmability and high-performance capabilities. To optimize resource utilization, both industry and academia have dedicated their efforts to enabling multi-tenant cloud FPGA environments [9]. In such FPGA platforms, a single FPGA instance is shared among multiple independent tenants simultaneously, significantly enhancing resource utilization efficiency. However, multi-tenant FPGA platforms expose victim tenants to various types of side channel attacks, particularly power side channels. [42], [48], [26], [40], [37], [58].

These power side channels leverage a crafted circuit to indirectly sense the voltage of another circuit sharing the same PDN with it. The key challenge is to design a sensitive circuit whose output is affected by the on-chip voltage and to co-locate it with the victim circuit. Compared with the above two side channels, the power side channel requires

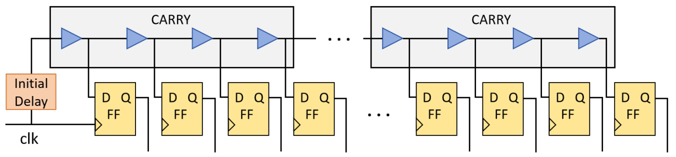


Fig. 2: CARRY-based circuit design. It requires a uniquely long chain of basic CARRY resources.

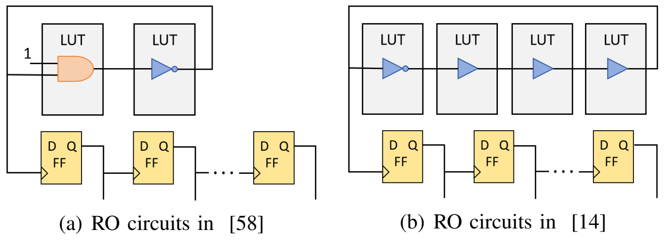


Fig. 3: Examples of RO circuit design. All of them contain a combinational loop where the signal propagates through a series of logic gates and is ultimately fed back to itself.

no additional assumption and has been frequently studied via extensive security implications, including extracting cryptographic keys [48], [26], [40], [37], [58], building covert channels [58], and stealing DNN model architectures [51], [57] or inputs [37]. As all these works abuse two essential FPGA computing structures, they are categorized as CARRY-based and LUT-based crafted circuits.

Firstly, CARRY-based circuits, also known as time digital converter (TDC)-based circuits [60], [40], [37], [51], utilize 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. When the voltage drops, specific bits in the outputs may flip, allowing the attacker to infer the voltage level by analyzing the number of bit flips. However, as shown in Figure 2, all of these CARRY-based circuits exhibit a unified architecture characterized by a long CARRY chain structure, making them easily identifiable by Design Rule Check [44], [15]. Further, these circuits incur a significant area overhead and are limited to vertical deployment, posing challenges for their implementation on multi-tenant FPGAs [36], [48].

Secondly, LUT-based works utilize LUTs to construct their on-chip sensors, including RO [58], VITI [48], PPWM [47], and 1LUTSensor [26]. Among these, only RO-based circuits are realistic design, as it does not require the details about the victim's circuit design and location. Specifically, each combinational loop contains an odd number of inverters implemented by LUTs, which can convert a binary input signal into its complement. The output of the last inverter is combinationally fed back into the input of the first inverter, with an inverse value. For instance, in Figure 3(a), the signal passes through an inverter, followed by an AND gate, and then is looped back to the input of the same inverter. In Figure 3(b), the signal goes through an inverter, followed by three buffers, and then returns to the input of the same inverter. When the attacker samples them at fixed time intervals, the increment can be employed to acquire power side channel information. However, because of its wide security implications, RO-based circuits have been disabled on commercial cloud services (e.g., AWS) by scanning the combinational loop structures [41], [30], [29].

The other LUT-based circuits including VITI [48], PPWM [47], and 1LUTSensor [26], unrealistically assume the victim's circuits in collusion with them in the calibration procedure. Consequently, attackers are required to acquire the hardware code of the victim circuits to perform offline calibration, which is nontrivial. Besides, they are all coarse-grained designs and their security implications are limited to extracting AES keys under specific assumptions about both the location and operating frequency of AES circuits. For instance, as demonstrated in [26], out of the six placements evaluated, the full key can only be extracted in a single placement.

Recently, Spielmann et al. utilize routing resources (e.g., wires) to craft Routing delay sensor (RDS). However, as the delay induced by wires is much smaller than the delay of basic computing resources (e.g., CARRY and LUT), its calibration is extremely complex. First, they need to use LUTs and CARRYs together to achieve a fine-grained configuration of initial delay. Second, they require the victim circuit to collude with their sensor to pick the most sensitive configuration for the given voltage fluctuations. To achieve this, the attacker is required to obtain the details about the victim's circuit design (i.e., hardware code) and location, which is unrealistic.

III. MUXLEAK

A. Threat Model and Assumptions

In this paper, we adopt the same threat model for multi-tenant cloud-FPGA used in related hardware security works [42], [48], [38], [26], [47], [57], as well as the FPGA virtualization works [34], [52], [27], [49]. Particularly, platform providers enforce a strict logic isolation by supplying each tenant with a physically exclusive FPGA region for their circuit deployment. To optimize power efficiency, these circuits use a shared on-chip PDN for power supply. With an FPGA region allocated, a tenant is allowed to define customized placements and routing constraints tailored to their designs, except for the malicious combinational loops that have been banned [41]. Different from [10], [46], we do not make the assumption that an FPGA chip is attached to a motherboard via PCIe or to a USB controller through a serial port.

B. Circuit Design

Aligned with previous FPGA-based power side channels [42], [48], [26], [47], [18], our proposed MUXLeak utilizes the correlation between signal propagation delay and supply voltage to deduce the power variations of the FPGA. Specifically, for the hardware circuits composed of CMOS transistors, their signal propagation delay is inversely related to on-chip supply voltage. When the supply voltage decreases, the signal propagation delay experiences a slight increase,

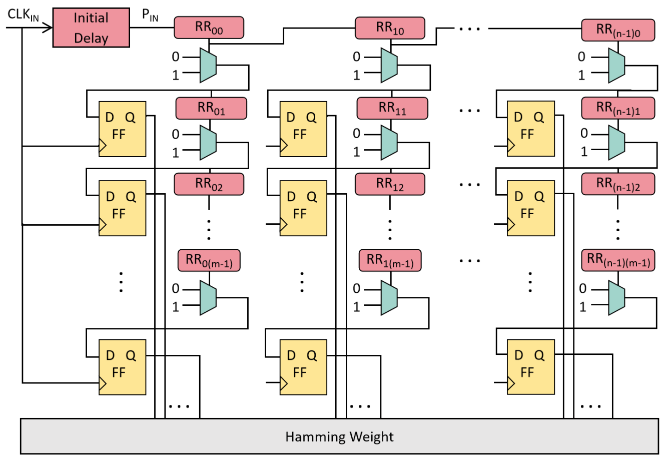


Fig. 4: An overview of MUXLeak.

whereas an excess supply voltage leads to a slight reduction in the signal propagation delay. While previous on-chip sensors typically exploit CARRYs or LUTs, they can be effectively mitigated by scanning the malicious use of these two resources. To this end, we carefully craft MUXLeak without any reliance on either LUT or CARRY.

Figure 4 depicts an overview design of MUXLeak, which consists of MUXes and FFs. We divide MUXes into n columns, each of which contains m MUXes. For each column, m MUXes are chained via a tapped delay line and the output of a preceding MUX is connected to the select signal of the next MUX. For each MUX, its select signal works with its two inputs (connected with '0' and '1', respectively) to decide its output. As the routing resource (e.g., wires) between every two MUXes is distinctive, we name each routing resource as $\texttt{RR}_{\texttt{ij}}$ $(0 \leq i \leq n-1,\ 0 \leq j \leq m-1)$, where i and j are used to identify their location within MUXLeak. Before reaching a FF, $\texttt{P}_{\texttt{IN}}$ is required to pass through a series of MUXes and RRs, which incur a delay to this signal propagation path. As MUXLeak contains different such signal propagation paths to generate a fine-grained scale of delay, when voltage fluctuates, these delays change differently.

To observe the variations in delay, these FFs are configured as positive edge-triggered and clocked by $\texttt{CLK}_{\texttt{IN}}$. In this way, their outputs change only when a rising edge of $\texttt{CLK}_{\texttt{IN}}$ arrives. Through these FFs, MUXLeak captures the propagation delay variations on FPGA by using a constant-phase $\texttt{CLK}_{\texttt{IN}}$ signal to sample the output signal of each MUX. By carefully configuring *Initial Delay* in the figure, the delay variation $(i.e., \Delta D)$ can cause the output of $\texttt{FF}_{\texttt{ij}}$ to flip from '1' to '0' if the following conditions are met:

$$(Z + 1/2) \times T < D(FF_{ij}) < (Z + 1/2) \times T + \Delta D, \quad (1)$$

where $Z \geq 0$, T is the period of the $\texttt{CLK}_{\texttt{IN}}$ signal, and $D(FF_{ij})$ is the overall delay of the propagation path from $\texttt{CLK}_{\texttt{IN}}$ to the FF located at $(Column_i,\ Row_j)$.

After deploying MUXLeak onto an FPGA board, the attacker first needs to calibrate the attack circuit by configuring the initial delay to make sure that the sensor output can consequently change when the on-chip voltage starts fluctuating. When the clock signal reaches its rising edge, MUXLeak samples all the outputs from FFs as its raw value, coined as *output*. Further, this raw value of one sampling is converted into a numerical value by calculating its Hamming weight [15], [42], termed as *readout*.

C. Sensing Voltage Fluctuation

In this section, we discuss how MUXLeak senses the voltage fluctuations at runtime. We further investigate factors that affect the sensitivity of MUXLeak. To achieve this, we need to establish the relationship between voltage fluctuations and delay variations. As the delay variations can induce bit flips in the MUXLeak's output, if there is a definite correlation between the voltage and delay, we can use the bit flips in MUXLeak's output to sense the voltage level.

As illustrated in Figure 4, the $\texttt{CLK}_{\texttt{IN}}$ signal goes through different paths to reach different FFs. To begin with, we define the delay of the propagation path from $\texttt{CLK}_{\texttt{IN}}$ to the FF located at $(Column_i,\ Row_j)$ using Equation 2:

$$D(FF_{ij}) = D_{\text{initial}} + \sum_{t=0}^{i-1} D(RR_{t0}) + \sum_{t=0}^{j} D(RR_{it}) + \sum_{t=0}^{j} D(MUX_{it}), \quad (2)$$

where $D(RR_{it})$ is the delay through the corresponding routing resource at $(Column_i,\ Row_t)$, $D(MUX_{it})$ is the delay of the MUX placed at $(Column_i,\ Row_t)$, and D_{initial} is a pre-configured delay. We note that the routing resource here specifically refers to the wires and switches connecting two CLBs, which is composed of CMOS logic and hence sensitive to voltage.

For each gate inside the circuits, a decrease in supply voltage induces an increase in signal propagation delay [60]. The delay of a gate can be approximated to be inversely proportional to its supply voltage:

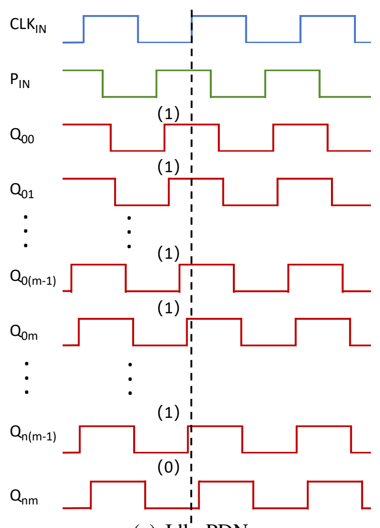
$$Delay = k \times Voltage^{-1}, \quad (3)$$

where $Delay$ can be any gate inside one circuit, $Voltage$ is the corresponding voltage, k is a constant.

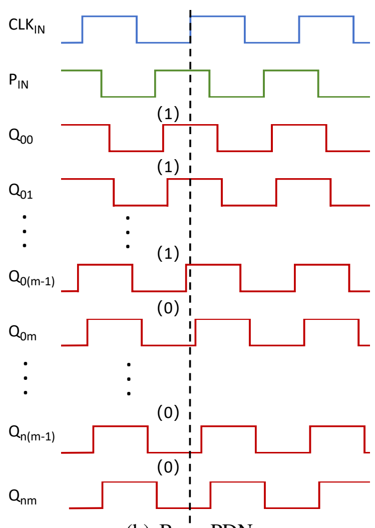
As the delay of each gate can be modeled to be approximately the same [58], we make the following approximation that the delay of each signal propagation path is also inversely proportional to its supply voltage:

$$D(FF_{ij}) \propto V_{ij}^{-1}. \quad (4)$$

As described in Section III-B, the delay variations can cause some bits of MUXLeak's output to flip. After establishing the relationship between voltage and delay in Equation 4, we can utilize the number of bit flips within MUXLeak's output to sense the voltage fluctuations. Figure 5 demonstrates an example timing diagram of MUXLeak, where the waveform of each FF has the same frequency but different phases. The waveforms in Figure 5(a) illustrate the behavior of MUXLeak when the PDN is idle, while the waveforms in Figure 5(b) illustrates the behavior for the case of a busy PDN. For each case, the vertical dotted line shows an instance of a rising edge



(a) Idle PDN



(b) Busy PDN

Fig. 5: An example timing diagram of MUXLeak. The impact of PDN instability is exaggerated to amplify the visual clarity.

on `CLK`$_{\texttt{IN}}$. Each `FF`$_{\texttt{ij}}$ captures the momentary value of `Q`$_{\texttt{ij}}$ at the first rising edge of `CLK`$_{\texttt{IN}}$ and outputs the captured value until the next rising edge on `CLK`$_{\texttt{IN}}$. By distinguishing whether the MUXLeak's output is '11...11...10' or '11...10...00', we can decide whether PDN is busy or idle.

Aligned with existing on-chip sensors [42], [58], there are two key characteristics that decide the MUXLeak's sensitivity: observable voltage range and granularity. Observable voltage range refers to the maximum voltage fluctuation that can be detected. Observable granularity refers to the minimum voltage fluctuation that can be detected. Both characteristics are crucial for MUXLeak to capture voltage fluctuations caused by a victim circuit. Here, we analyse the factors that affect the two characteristics.

Observable voltage range: For each FF, the observable range depends on the difference between the maximum and minimum voltages that can be detected by MUXLeak. If we approximately assume the voltage of each FF is equal, the observable voltage range can be represented as:

$$W_{\triangle} \propto \max|D(F_{ij})-D(F_{00})|,\ 0 \le i \le m-1, 0 \le j \le n-1. \quad (5)$$

Given that the MUXes on the same FPGA board share the same production process and materials, we approximate their delays to be equal, denoted as $D(MUX)$. By substituting Equation 2 into Equation 5, we can express $W_{\triangle}$ as:

$$W_{\triangle} \propto \max|\sum_{t=1}^{i} D(RR_{t0}) + \sum_{t=1}^{j} D(RR_{it}) + j \times D(MUX)|, \\ 0 \le i \le m-1, 0 \le j \le n-1. \quad (6)$$

When and only when $j = n-1$, $W_{\triangle}$ reaches its maximum. Considering the delay of MUXes (which is at the granularity of nanosecond) is much higher than the delay of routing resources, the observable voltage window is primarily determined by n.

Observable granularity: The granularity of MUXLeak is typically expressed as the minimum voltage change that can be detected. We express it as Equation 7:

$$G \propto \min|D(F_{ij}) - D(F_{kl})|, 0 \le i,k \le m-1; 0 \le j,l \le n-1. \quad (7)$$

By substituting Equation 2 to Equation 7, G can be expressed as:

$$G \propto \min|\sum_{t=0}^{i} D(RR_{tj}) + \sum_{t=1}^{j} D(RR_{it}) - \sum_{t=0}^{k} D(RR_{tl}) - \sum_{t=1}^{l} D(RR_{kt}) \\ + (j-l) \times D(MUX)|, 0 \le i,k \le m-1; 0 \le j,l \le n-1. \quad (8)$$

When and only when $j = l$, G reaches its minimum. Since the delay of the routing resource is proportional to the length of the wire and the area of the circuit, we use the routing length to control its delay. Ideally, within a specified observable range (n), the variations in routing lengths to reach the FFs placed in the same row should be evenly increased. Therefore, the granularity depends on m, which determines the variety of delays of routing resources. A larger m can result in better granularity.

D. Calibration

The goal of the calibration is to ensure that the bit-flip number from MUXLeak's output changes accordingly when voltage supply starts fluctuating. As shown in Figure 6(a), for the uncalibrated MUXLeak, although there is a delay variation in a busy PDN, it cannot cause a bit flip for MUXLeak's output (i.e., `Q`$_{\texttt{ij}}$). As such, MUXLeak without calibration is insensitive to voltage fluctuations. To ensure and optimize MUXLeak's sensitivity across its different placements on different FPGA boards, we propose a calibration design for MUXLeak. After deploying MUXLeak onto the board, the

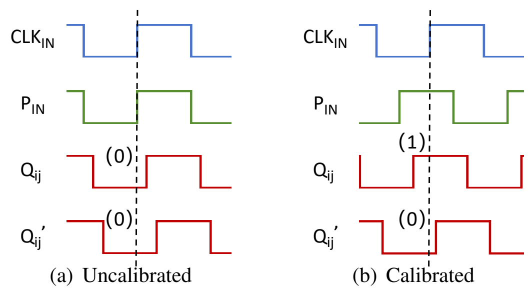


Fig. 6: An example timing diagram of uncalibrated/calibrated MUXLeak. The vertical dotted line shows a rising edge of the $\texttt{CLK}_{\texttt{IN}}$. $\texttt{Q}_{\texttt{ij}}$ denotes the output of $\texttt{FF}_{\texttt{ij}}$ in an idle PDN, while $\texttt{Q}_{\texttt{ij}}{}'$ denotes the same output in a busy PDN.

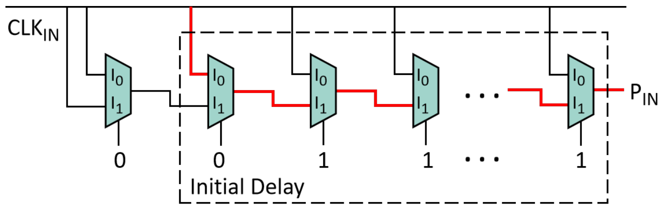


Fig. 7: MUX-based calibration design. The red line denotes the actual signal propagation path. When the last c select signals are configured as '1', $\texttt{CLK}_{\texttt{IN}}$ is required to go through $c+1$ MUXes (within the dashed box) before reaching $\texttt{P}_{\texttt{IN}}$.

attacker can use this design to dynamically adjust the initial delay. As shown in Figure 6(b), compared to the uncalibrated MUXLeak, a calibrated MUXLeak with a configured initial delay, has more flippable outputs (e.g., $\texttt{Q}_{\texttt{ij}}$) for the given voltage fluctuations.

Previous realistic attacks [14], [51], [40], [37] rely on the well-studied basic resources that are in the spotlight of existing defenses [18], [42] (i.e., LUTs and CARRYs) to generate the initial delay. Their key observation is that CARRY contains an enable signal that can dictate the number of units involved in the function, while LUTs can be carefully configured to enable a similar signal. To this end, they chain the LUTs or CARRYs as a tapped delay line and then dynamically adjust the initial delay via configuring its enable signals.

After observing that a MUX can dynamically change the signal propagation path via configuring its select signal, we propose a calibration design with MUXes only. Figure 7 shows our MUX-based design. To chain these MUXes to generate initial delay, each MUX is configured such that its $\texttt{I}_0$ is connected to $\texttt{CLK}_{\texttt{IN}}$, while its $\texttt{I}_1$ is connected to the output of the preceding MUX in the sequence. The select signal is employed to control the signal flow, and it is composed of an ordered sequence ranging from 0 to 1. As indicated by the red line, when the last c select signals are set to '1', $\texttt{CLK}_{\texttt{IN}}$ is required to go through $c+1$ MUXes (within the dashed box) before reaching $\texttt{P}_{\texttt{IN}}$.

We adopt the procedures established by previous calibrations [18], [51] to configure the initial delay for MUXLeak. First, we develop a C-based driver program executed on the CPU, which can dynamically change the select signal. Second, we iteratively test all possible values for the number of activated MUXes via different select signals. Last, the aim of our calibration is to select an initial delay that maximizes the number of outputs satisfying Equation 1. Consequently, we pick the configuration with the closest number of '1' in the output value to $(m \times n)/2$ when PDN is idle as the optimal configuration. In this way, the initial delay can be determined without any involvement of victim circuits.

E. MUXLeak Implementation

MUXLeak is constructed using $m \times n$ MUXes and FFs. Though MUXLeak can be compiled with an arbitrarily chosen m and n, the optimal size is influenced by the following two factors.

Sensitivity: As we discussed in Section III-C, the observable voltage range of MUXLeak mainly depends on the number of MUXes in each column (m) and the observable granularity mainly depends on the number of columns (n). To better sense the voltage fluctuations, these two values should be as large as possible.

Resource budget: In the multi-tenant FPGAs, the financial cost associated with MUXLeak is directly proportional to its consumption of FPGA resources. Besides, the attacker is restricted to a specific FPGA area with limited resources. To deploy MUXLeak onto a victim FPGA platform, the resource utilization should be limited at the same level with existing on-chip sensors.

Determining m and n: Aligned with TDC [18], we select 128 FFs to implement MUXLeak. Correspondingly, 128 MUXes are employed. To group these MUXes, we have all possible pairs of (m, n) and select $m = 4, n = 32$ for MUXLeak. Our MUXLeak achieves an average delay difference between two adjacent MUXes in each column to be (0.904±0.154) ns, and the Kurtosis of delay difference between each two FFs is 7.81, indicating that these delays are not absolutely uniform but acceptable.

To precisely map our design to a victim FPGA board, we directly use MUXF7 primitives to instantiate the MUXes. The implementation of edge-triggered flip-flops employs FD primitives. The MUXes in the same column are connected by using the output of the previous MUX as the select signal input for the next MUX. Last, on the SoC platforms, we utilize AXI-4 buses to establish an interface between the ARM processor and the TDC output and develop a C-based driver that utilizes mmap system call to obtain the readouts. On the custom built FPGA-based platforms, the communication between FPGA and processor is achieved via UART.

Table I lists and compares the FPGA resource utilization of our implementations of MUXLeak with existing realistic on-chip sensors [60], [40], [58], [14]. All these results are obtained from Vivado Floorplanning interface. As described in Section III-D, we only use MUX for calibration. With 32 MUXes, the MUXLeak can be well calibrated. To make the chains within MUXLeak has more diverse routing path, we restrict MUXLeak to a $\texttt{Pblock}$ having $\approx$ 1.5× MUXes as required (i.e., 120 slices). This intentional over-estimation facilitates in optimizing the placement and routing (P&R) process using the Vivado toolkit.

TABLE I: A comparison of MUXLeak and mainstream on-chip sensors regarding their resource usage.

On-chip Sensor	Calibration			Sensing Resources			
	CARRY	LUT	MUX	LUT	CARRY	MUX	FF
RO in [14]	0	0	0	20	0	0	160
RO in [58]	0	0	0	40	0	0	80
TDC in [60])	4	32	0	0	32	0	128
TDC in [40]	0	18	0	0	16	0	64
MUXLeak	0	0	**32**	0	0	**128**	128

IV. Evaluation

In this section, we first evaluate MUXLeak's capabilities, and then apply MUXLeak to mount two side-channel attacks.

Experimental setup: We evaluate MUXLeak on multiple machines across various FPGA architectures summarized in Table II, including two System-on-Chip (SoC) boards and one custom-built machine. To build the custom-built machine, the Basys 3 board is connected to a Huawei MateBook 14 laptop via UART and powered through the USB interface. To compile a C-based program, gcc 7.5.0 is used. To deploy hardware code onto these FPGAs, we use Vivado version 2019.1 for synthesis, implementation, and bitstream generation.

A. Characterizing MUXLeak

To show the sensitivity of MUXLeak to voltage fluctuations, we opt for power virus circuits [20] as a victim to generate varying voltage fluctuations, aligned with [58]. The power virus circuit consists of a number of instances that are implemented as RO circuits. Similar to the RO circuit shown in Figure 3(a), within the power virus circuits, each RO circuit consists of a single inverter, an AND gate, and an FF. The operation of the inverter can be 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.

We compare MUXLeak's sensitivity with that of the state-of-the-art sensor (i.e., TDC [60]) on the PYNQ-Z2 board. First, we implement MUXLeak as described in Section III-E. Second, we follow [60]) to reproduce a TDC circuit and its resource usage is shown in Table I. We then deploy MUXLeak (resp., TDC) via the same sampling frequency (200 MHz) and placement restriction (i.e., `Pblock`).

MUXLeak's sensitivity under different victim activities: Our first experiment is to characterize the readout change of MUXLeak to various victim activities. Specifically, we create 16,000 instances of power viruses to cover over 60% routing places of our PYNQ board and activate about 40% of available LUT resources (16,000 instances is a high workload that unlikely occurs for normal circuits). We then divide them into 32 groups and each group has 500 evenly-distributed

TABLE II: System configurations.

Property	PYNQ-Z2	ALINX AXU3EGB	Basys 3
FPGA Family	Zynq-7000	Zynq UltraScale+ MPSoC	Artix-7
Number of MUXes	39,900	61,740	15,600
Logic Cells	85,000	154,350	33,280
Clock Frequency	125 MHz	200 MHz	100 MHz
FPGA Core Voltage	0.95∼1.0 5V	0.825∼0.876 V	0.95∼1.05 V
CPU Model	Cortex-A9	Cortex-A53	i7-10510U
Memory (DRAM)	512 MB	5 GB	16 GB
OS	Ubuntu-18.04	Ubuntu-18.04	Ubuntu-20.04

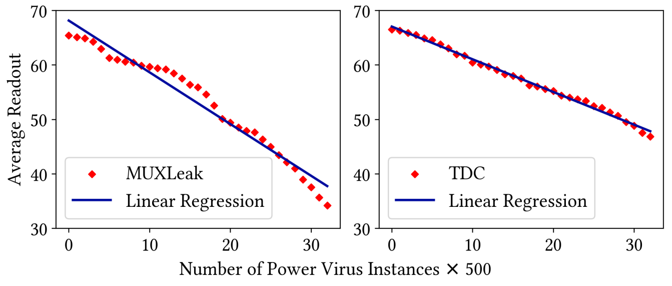
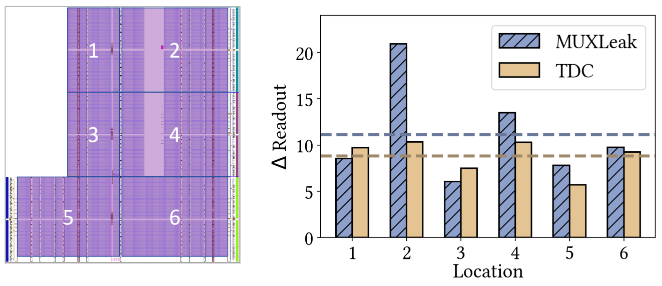


Fig. 8: The sensitivity of MUXLeak and TDC [18] in the same placement under different victim activities.

instances. When we activate a varying number of these groups, the voltage varies from 33 levels. For each level, we collect 100,000 readouts from MUXLeak (resp., TDC) 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.

Figure 8 shows the relationship between these readouts and the number of activated power virus instances. These two values of MUXLeak exhibit a roughly linear relationship with a Pearson correlation coefficient of -0.981, while TDC achieves -0.994. 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 MUXLeak and can be accurately estimated using its readout. Furthermore, compared with the most sensitive on-chip sensor (i.e., TDC), MUXLeak achieves a high regression coefficient of -0.94, while TDC achieves -0.60. This indicates that MUXLeak achieves the same order of granularity under the given sensor placement.

MUXLeak'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 MUXLeak 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 9(a)). Subsequently, we put MUXLeak (resp., TDC) in each of the 6 FPGA regions via a `Pblock` constraint with all the 8,000 power virus instances off and on.



(a) (b)

Fig. 9: The sensitivity of MUXLeak and TDC [18] under different placements. The index as the x-axis in Figure 9(b) corresponds to the region index in Figure 9(a). The dashed line indicates the average value of a sensor.

We sample 100,000 readouts under each setting and calculate their average.

As shown in Figure 9(b), MUXLeak successfully senses the given voltage fluctuation under all the above placements. Specifically, MUXLeak achieves the best performance when it is placed closest to the victim circuit (i.e., region 2). Further, when MUXLeak is placed under the worst-case placement (i.e., region 3 or region 5), its readouts are still sensitive to the voltage fluctuations. Last, compared with TDC, MUXLeak achieves the same level of observable granularity under all placements.

B. Extracting Cryptographic Keys

AES is the most widely adopted symmetric-key cryptographic algorithm, included in almost all modern crypto libraries. It operates on fixed-length data blocks and offers key sizes of 128, 192, or 256 bits, providing versatility for various applications. Today, AES has been implemented in different devices, including hardware circuits on FPGA platforms [4], [21], [7].

Prior work has demonstrated that a remote attacker can extract the full secret key of AES circuits via Correlation Power Analysis (CPA) attacks [40], [15], [45], [11]. These attacks leverage the Pearson Correlation Coefficient [8] 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 utilize 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 true key value. The number of traces used by attackers depends on the resolution of their side channel, ranging from 1,000 [11] to more than 1,000,000 [48].

Experimental setup: We implement an open-source AES-128 module [4] that co-resides with MUXLeak on the Basys3 board. Besides, we re-implement TDC for extracting the AES key as a comparison. By default, the clock frequencies of the attack circuit and the AES are set to 200 MHz and 20 MHz, respectively. When investigating the impact of AES frequency on our attack, the maximum frequency for AES is set to 100 MHz, as the clock frequency of Basys3 is 100 MHz, as shown in Table II.

For each attack, we collect 100,000 power side-channel traces. In each trace acquisition, we transmit the key `K` and the plaintext `PT` to the AES core, and record the MUXLeak'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. Please note that in practical attacks, attackers can resort trace alignment and automatic triggering techniques as described in [40] to avoid reliance on this signal. To evaluate the performance of MUXLeak, we run the key extraction attack under 8 placements and 4 different AES clock frequencies. Under each setting, we repeat the experiments five times (each time with a different key) to ensure more robust results, aligned with [42]. The keys and plaintexts employed in this evaluation are listed in Table III.

We opt for the key rank (KR) estimation metric, which assesses the number of key candidates the attacker needs to test [17]. Due to the bounded error in existing key rank estimation algorithms, the key rank is reported as a tight range with an upper bound and a lower bound. For example, if an attacker has no side-channel information, then the upper key rank equals the entire key space, i.e., 2^{128} in the case of AES-128. Alternatively, when the entire key is broken, the lower bound of the KR drops to unity and the upper bound of the KR tightly bounds the range, signaling the correct key is recoverable with a negligible number of guesses. In this situation, attackers can sort each guessed byte value by correlation coefficient and thus acquire the true AES key. To save the computation time, we use the open-source analyzing tool [11] to run our CPA analysis on an NVIDIA GeForce RTX 3090 GPU.

Experimental results: Figure 10 shows the average upper and lower bounds of the key rank estimation metric to evaluate the impact of the placement of MUXLeak. Figure 10(a) is a color gradient that rates all 8 experiment locations based on their key rank with 20k power traces, revealing that the efficacy of a power analysis attack can be deduced to be reliant on the placement of MUXLeak on the FPGA. This observation aligns with the findings reported in [48], [42], where the location-dependent sensitivity is attributed to the non-uniformity of the PDN across the FPGA board. Moreover, Figure 10(b) shows the key rank variability of MUXLeak

TABLE III: Key and plaintext values used in the evaluation.

Secret Key	Key Value	Plaintexts
Key1	`0x7d266aecb153b4d5d6b171a58136605b`	
Key2	`0xe3fb107fa4aaeb7130f411d4c88dbf6c`	
Key3	`0xa89e2fd6926dc2478402b717631d08ce`	$PT_0 = 0$ $PT_{i+1} = CT_i$
Key4	`0xa3a03d60c06457dc65d8afd5815f629c`	
Key5	`0xe1055ac2abadea4fc7fc6be1310448d9`	

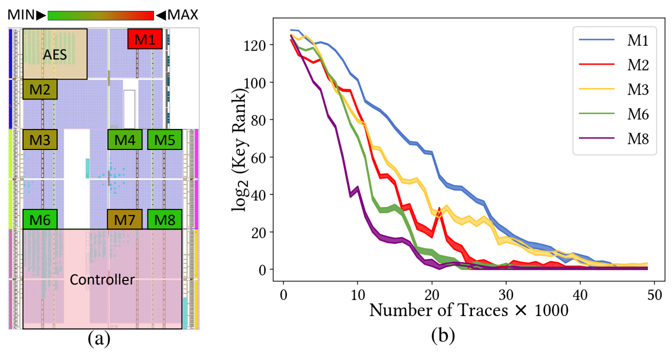


Fig. 10: Key rank estimation for MUXLeak. The curves in Figure 10(b) are named after the number on each placement shown in Figure 10(a).

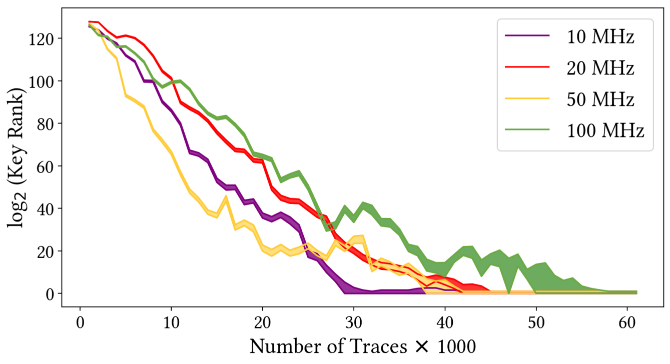


Fig. 11: The impact of the AES frequency on our attack. MUXLeak has the capability to expose the entire AES key within 60k traces across various operating frequencies under the worst placement, i.e., M1 in Figure 10(a).

TABLE IV: Number of traces required to break the full AES 128-bit key for different sensor placements. MUXLeak just requires 20k-45k traces to extract the full key.

Setting	**Traces to Extract Full Key** (×1000)								
	M1	M2	M3	M4	M5	M6	M7	M8	Average
MUXLeak	**45**	**25**	**41**	**37**	40	**35**	**28**	**20**	**33.9**
TDC [18]	48	60	45	63	**35**	46	38	50	48.2

across 5 selected placements out of the total eight, 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 10(a). The full results are summarized in Table IV. Outperforming the TDC, MUXLeak just requires 20 k-45 k traces to extract the full key.

We then evaluate the impact of the AES frequency on our attack, utilizing the worst-case placement for attackers (i.e., M1). Figure 11 illustrates the corresponding key rank outcomes. Notably, even with the worst-case placement, MUXLeak has the capability to expose the entire AES key within 55k traces across various operating frequencies from 10 MHz to 100 MHz. Furthermore, we note that the performance of the attack is dependent on the frequency. When the frequency is set to 10 MHz, a mere 30k traces are required to recover the full key. Under all the above settings, the time for extracting the full AES key ranges from 0.93 to 2.54 hours.

C. Stealing DNN Model Architectures

To speed up the inference process and reduce energy consumption, FPGA-based DNN accelerators have been widely deployed in high-performance cloud computing platforms [38], [57]. Compared to CPUs and GPUs, FPGAs have significant advantages in performance and power consumption for executing model inferences [35], [53]. For instance, in a standard MLPerf [39] benchmark, AMD Xilinx's high-end AI Adaptive Compute Acceleration Platform (ACAP) VCK5000 achieves 1.8× frames per second per watt compared to the Nvidia Ampere flagship GPU (i.e., A100 SXM).

However, due to the valuable and confidential nature of their optimized architectures which require significant time and effort to develop, these DNN accelerators have been the target of remote attackers [57], [51], [37]. Furthermore, knowledge of a specific DNN architecture can facilitate many other adversarial attacks such as adversarial example attacks and model extraction attacks. The success rate of such attacks is highly dependent on the similarity between the target victim model and the substitute model [12], [54]. Our key insight of using MUXLeak to steal DNN architectures from an FPGA-based accelerator is that the model inference is performed in layer order. As each layer runs, the voltage will fluctuate by different magnitudes. Attackers can exploit MUXLeak to capture and then analyze this information.

To mount a model architecture stealing attack via our MUXLeak side channel, we assume that a MUXLeak circuit is co-located with the victim FPGA accelerator circuit. The attacker's objective is to recover the architecture of the co-located DNN model through the MUXLeak side channel. As shown in Figure 12, the attack has two phases: an *offline training* phase and an *online attack* phase. In the *offline training* phase, we collect a sufficient number of MUXLeak traces to construct a well-trained sequence-to-sequence classifier, which can translate a MUXLeak trace to its corresponding layer types. In the *online attack* phase, we either query the black-box target model or wait for its inference process to initiate, during which we utilize the co-located MUXLeak circuit to gather the corresponding MUXLeak trace. We then employ the offline-trained classifier to recover the victim model structure.

Experimental setup: We carry out our experiments on our PYNQ-Z2 board. The victim DNN model is implemented by an open-source framework [57]. To collect the dataset, we generate 120 different random models and deploy them separately on the FPGA board. These models have random numbers (in the range of [4, 16]) and types of network layers, including various model architecture families (e.g., AlexNet, VGG, and

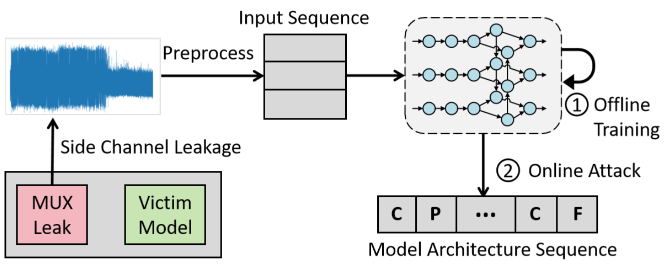


Fig. 12: MUXLeak-based model architecture stealing.

random architectures). The model size is constrained by the FPGA logic resources. We collect 40 traces for each model and split the traces into 10 folds and select one fold as the test set. The input size used by these models is assumed as 3×64×64. The models are pre-trained on Tiny ImageNet-200 dataset, which comprises 200 image classes. The collected trace is a sequence in which the MUXLeak readouts are arranged in the temporal order, which represents the switching activities on the FPGA at different moments.

We use a Transformer model [51] as the classifier to translate a side channel trace to corresponding model layer sequence, which adopts a single-layer encoder and a single-layer decoder. This model utilizes weight sharing between the decoder embedding and the decoder projection to enhance generalization. We use the Adam optimizer with OneCycleLR scheduler to train the classifier and choose the cross-entropy loss function to evaluate its performance. The classifier is trained in a machine with an NVIDIA GeForce RTX 3090 GPU (24 GB video memory), Intel i9-10920X CPU (24 logical cores) and 128 GB DRAM memory. The deep learning framework used here is PyTorch 2.0.1 with Python 3.8.13. Aligned with previous work [12], [55], we select Levenshtein Distance Accuracy (LDA) as a metric to evaluate the structure recovering performance, which represents the similarity between a predicted structure and a ground-truth structure.

Experimental results: Table V reports the LDA for different types of layers when the embedding dimension is set as 128, 256, 512, and 1024 respectively. We observe that the Transformer with the dimension of 512 achieves the best performance of 91.1%. Table VI shows the predicted layer sequences of AlexNet and VGG11 when selecting the embedding dimension as 512. As each model architecture is tested for 40 traces, for the predicted sequence, we show one of the bad cases, and the accuracy is averaged across all 40 traces. We observe that the incorrect layer is more likely to appear in the latter half. We believe that this may be due to the smaller data size in the latter half of the layer, resulting in less significant changes in power consumption and shorter computation time for these layers.

V. Discussion

Hiding the power consumption patterns: As our MUXLeak relies on the precise side channel information obtained from a shared PDN, a straightforward mitigation is to eliminate the voltage fluctuations with the improvements in the PDN.

TABLE V: Classification accuracy (%) of network steps with different embedding dimensions.

Dimension	Convolutional	Pooling	Fully-connected	Overall
128	83.2	87.8	98.3	87.3
256	85.3	86.6	97.8	87.8
512	88.2	90.3	99.1	91.1
1024	86.1	86.3	97.8	88.2

TABLE VI: The layers of profiled models (selected from 12 tested models). C = Convolutional Layer, P = Pooling Layer, F = Fully-connected Layer.

Model	Original Layer Sequence Predicted Layer Sequence	Accuracy
AlexNet	C-P-C-P-C-C-C-P-F-F-F C-P-C-P-C-C-C-C-P-F-F-F	89.4%
VGG11	C-P-C-P-C-C-P-C-C-P-C-C-P-F-F-F C-P-C-P-C-C-P-C-C-C-P-F-F-F	90.3%

However, while PDN aims to maintain a stable voltage supply to each tenant circuit, it still cannot eliminate variations in current [59], [22]. The inherent challenges of optimizing power distribution for diverse and unpredictable workloads [43], coupled with the necessity for innovative isolation and stabilization solutions [50], render this problem inherently complex and time-consuming to address.

An alternative approach is to inject artificially-introduced random noise to PDN to hide the power consumption patterns of victim circuits. To achieve this, there has two approaches for developers. First, they can modify their functional logic design to construct a constant-power implementation, thereby masking the power consumption of every operation. However, such modification requires developers to invest significant effort, especially for the large-scale projects. Second, developers can deploy a fence circuit [16], [28], [23] to isolate the victim circuit from the attacker-crafted circuit. These fence circuits are placed surrounding the victim circuits to dynamically inject noise to PDN. However, to effectively conceal the power consumption patterns caused by the victim circuit, the fence circuits are required to incur non-negligible execution overhead. Policies that can introduce such distortions with minimum performance overhead is non-trivial.

Preventing crafted circuits from co-residing with victim: As existing power side channel attack relies on physically residing a crafted circuit with the victim circuits on the same FPGA board, an effective way to mitigate this risk is to prevent the attacker from co-locating her circuit with the victim. To achieve this, users can purchase the ownership of the whole FPGA board as well as the hosting server, which can eliminate the threat of side-channel-related attacks induced by sharing FPGA resources. However, this approach increases the costs of deploying FPGA circuits for cloud service users. Alternatively, cloud service providers can improve their scheduling policy to scatter the circuits from different users or companies on the cloud, which can effectively decrease the chance of attacker's circuits co-locating with the victim.

Developing effective DRC tools: Besides the co-residence prevention above, several design rule checking (DRC) tools have been proposed to detect malicious circuits by analyzing bitstreams, RTL designs, or netlists [3], [44], [30], [29], [19]. In practice, cloud providers may deploy such tools to inspect tenant submissions and block circuits containing suspicious structures. Since MUXLeak is the first to exploit MUXes to craft malicious circuits for power side channels, it is believed to bypass current DRC tools focused on detecting attacks involving LUTs and CARRY chains. However, MUXLeak's highly regular structure of chained MUXes may present a clear fingerprint that pattern-based detection tools could leverage. This underscores an ongoing arms race between malicious design techniques and detection mechanisms. We frame the development of effective DRC tools tailored to detect MUXLeak as an open research direction.

Testing MUXLeak on non-Xilinx platforms: Aligned with prior FPGA power side-channel studies [58], [57], [56], [42], we evaluate MUXLeak on three Xilinx FPGA-based platforms, including the Xilinx UltraScale architecture used in AWS EC2 (specifically, the Xilinx UltraScale+ VU9P FPGAs [5]). We focus on Xilinx devices because it is the largest FPGA vendor [1] and provides comprehensive documentation. While MUXes are also present in other non-Xilinx FPGA platforms (e.g., Intel [2]), substantial differences in logic and routing architectures may impact the feasibility of MUXLeak. Thus, applying MUXLeak and other FPGA power side-channel techniques to non-Xilinx platforms remains an important direction for future work.

VI. Conclusion

In this paper, we reveal MUXLeak, a novel on-chip sensor that exploits MUXes to craft a stealthy power side channel on multi-tenant FPGAs. Our key observation is that fine-grained power side channel information can be leaked via the delay variations of MUXes, which exhibit an inverse relationship with on-chip voltage. By establishing carefully designed connections, we can measure these delay variations by monitoring the number of bit flips in MUXLeak's output. To demonstrate the viability of MUXLeak, we perform a comprehensive evaluation of MUXLeak on three FPGA-based platforms. First, we show that MUXLeak achieves comparable sensitivity with TDC (i.e., the most sensitive crafted circuit until now). We then leverage MUXLeak to mount realistic power side channel attacks, successfully extracting AES keys within 2.54 hours and stealing DNN model architectures with an accuracy of over 90%.

VII. Acknowledgements

This work was supported by the National Key R&D Program of China (No. 2022YFB2703301), the Science and Technology Development Program of Two Districts in Xinjiang, China under Grant No. 2024LQ03004, the Australian Research Council (FL240100217). Jinhua Cui's work was supported by the National Natural Science Foundation of China (Grant No. 62402169), the Key Research and Development Program of Hunan Province (Grant No. 2024JK2011), the CCF-Huawei Populus Grove Fund (CCF-HuaweiTC202402).

References

[1] "Field-programmable gate array (fpga) market (2024-2032): Growth forecast," https://www.linkedin.com/pulse/field-programmable-gate-array-fpga-market-2024-2032-14kuf/, 2024.

[2] "Arria 10 core fabric and general purpose i/os handbook," https://www.intel.com/content/www/us/en/docs/programmable/683461/current/alm-resources.html, 2025.

[3] Q. A. Ahmed, T. Wiersema, and M. Platzner, "Proof-carrying hardware versus the stealthy malicious lut hardware trojan," in *International Symposium on Applied Reconfigurable Computing*. Springer, 2019, pp. 127–136.

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

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

[6] C. Chiasson and V. Betz, "Coffe: Fully-automated transistor sizing for fpgas," in *International Conference on Field-Programmable Technology (FPT)*, 2013, pp. 34–41.

[7] P. Chodowiec and K. Gaj, "Very compact fpga implementation of the aes algorithm," in *Cryptographic Hardware and Embedded Systems*, 2003, pp. 319–333.

[8] 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.

[9] 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, pp. 487–506.

[10] C. Fang, N. Miao, H. Wang, J. Zhou, T. Sheaves, J. M. Emmert, A. Sasan, and H. Homayoun, "Gotcha! i know what you are doing on the fpga cloud: Fingerprinting co-located cloud fpga accelerators via measuring communication links," in *ACM SIGSAC Conference on Computer and Communications Security*, 2023.

[11] 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, pp. 501–506.

[12] Y. Gao, H. Qiu, Z. Zhang, B. Wang, H. Ma, A. Abuadbba, M. Xue, A. Fu, and S. Nepal, "Deeptheft: Stealing dnn model architectures through power side channel," in *IEEE Symposium on Security and Privacy*, 2023.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] C. Glowacz, V. Grosso, R. Poussier, J. Schüth, and F.-X. Standaert, "Simpler and more efficient rank estimation for side-channel security assessment," in *International Workshop on Fast Software Encryption*, 2015, pp. 117–129.

[18] D. R. E. Gnad, C. D. K. Nguyen, S. H. Gillani, and M. B. Tahoori, "Voltage-based covert channels using fpgas," *ACM Transactions on Design Automation of Electronic Systems*, 2021.

[19] 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*, 2018, pp. 286–289.

[20] D. R. 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).* IEEE, 2017, pp. 1–7.

[21] T. Good and M. Benaissa, "Aes on fpga from the fastest to the smallest," in *Cryptographic Hardware and Embedded Systems*, 2005, pp. 427–440.

[22] 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, 2018.

[23] 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*, 2007, pp. 281–295.

[24] A. Inc., "7 series fpgas configurable logic block: User guide (ug474)," 2016. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

[25] ——, "Ultrascale architecture configurable logic block: User guide (ug574)," 2017. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug574-ultrascale-clb.pdf

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

[27] 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.

[28] 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.

[29] 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*, pp. 1–26, 2019.

[30] T. M. La, K. Matas, N. Grunchevski, K. D. Pham, and D. Koch, "Fpgadefender: Malicious self-oscillator scanning for xilinx ultrascale+ fpgas," *ACM Transactions on Reconfigurable Technology and Systems*, pp. 1–31, 2020.

[31] 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, p. 818–831.

[32] J. Liu, S. Zeng, L. Ding, W. Soedarmadji, H. Zhou, Z. Wang, J. Li, J. Li, Y. Dai, K. Wen, S. He, Y. Sun, Y. Wang, and G. Dai, "Flightvgm: Efficient video generation model inference with online sparsification and hybrid precision on fpgas," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2025, p. 2–13.

[33] Y. Luo, A. S. Rakin, D. Fan, and X. Xu, "Deepshuffle: A lightweight defense framework against adversarial fault injection attacks on deep neural networks in multi-tenant cloud-fpga," in *IEEE Symposium on Security and Privacy*, 2024, pp. 34–34.

[34] 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.

[35] Y. Ma, Y. Cao, S. Vrudhula, and J.-S. Seo, "Automatic compilation of diverse cnns onto high-performance fpga accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, pp. 424–437, 2020.

[36] S. Moini, X. Li, P. Stanwicks, G. Provelengios, W. Burleson, R. Tessier, and D. Holcomb, "Understanding and comparing the capabilities of on-chip voltage sensors against remote power attacks on fpgas," in *International Midwest Symposium on Circuits and Systems*, 2020, pp. 941–944.

[37] 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, pp. 1639–1644.

[38] A. S. Rakin, Y. Luo, X. Xu, and D. Fan, "{Deep-Dup}: An adversarial weight duplication attack framework to crush deep neural network inmulti-tenant fpga," in *USENIX Security Symposium*, 2021, pp. 1919–1936.

[39] 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.

[40] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on fpgas," *IEEE Design & Test*, pp. 58–66, 2021.

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

[42] D. Spielmann, O. Glamočanin, and M. Stojilović, "Rds: Fpga routing delay sensors for effective remote power analysis attacks," *Cryptographic Hardware and Embedded Systems*, pp. 543–567, 2023.

[43] 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.

[44] 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*, 2019.

[45] T. Swamy, N. Shah, P. Luo, Y. Fei, and D. Kaeli, "Scalable and efficient implementation of correlation power analysis using graphics processing units (gpus)," in *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy*, 2014, pp. 1–8.

[46] S. Tian, I. Giechaskiel, W. Xiong, and J. Szefer, "Cloud fpga cartography using pcie contention," in *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2021, pp. 224–232.

[47] 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.

[48] ——, "Viti: A tiny self-calibrating sensor for power-variation measurement in fpgas," *Cryptographic Hardware and Embedded Systems*, pp. 657–678, 2022.

[49] 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.

[50] 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.

[51] X. Yan, X. Lou, G. Xu, H. Qiu, S. Guo, C. H. Chang, and T. Zhang, "Mercury: An automated remote side-channel attack to nvidia deep learning accelerator," in *International Conference on Field Programmable Technology*, 2023, pp. 188–197.

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

[53] C. Zhang, S. Cao, G. Dai, C. Geng, Z. Yao, W. Xiao, Y. Liu, M. Wu, L. Zhang, G. Sun, Z. Ji, R. Wang, and R. Huang, "Fine-grained structured sparse computing for fpga-based ai inference," *IEEE*

Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2024.

[54] 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.

[55] X. Zhang, Z. Zhang, Q. Shen, W. Wang, Y. Gao, Z. Yang, and J. Zhang, "Segscope: Probing fine-grained interrupts via architectural footprints," in *High Performance Computer Architecture*, 2024.

[56] X. Zhang, J. Zou, Y. Yang, Q. Shen, Z. Zhang, Y. Gao, Z. Wu, and T. Carlson, "LeakyDSP: Exploiting Digital Signal Processing Blocks to Sense Voltage Fluctuations in FPGAs," in *ACM/IEEE Design Automation Conference (DAC)*, 2025.

[57] 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*, pp. 4377–4388, 2021.

[58] M. Zhao and G. E. Suh, "Fpga-based remote power side-channel attacks," in *IEEE Symposium on Security and Privacy*, 2018, pp. 229–244.

[59] H. Zhu, W. Cao, and X. Zhang, "Pdnsig: Identifying multi-tenant cloud fpgas with power distribution network-based signatures," in *IEEE/ACM International Conference on Computer Aided Design*, 2023, pp. 1–8.

[60] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in fpgas," in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, 2013, pp. 101–104.

Xin Zhang received the B.Sc. degree in information security from Hunan University in 2022. He is currently pursuing the Ph.D. degree with Peking University, Beijing, China. His research interests include system security, computer architecture and side channel attack.

Jiajun Zou received the B.Sc. degree in software engineering from Hunan University in 2024. He is currently pursuing the Master degree with Peking University, Beijing, China. His research interests include system security and side channel attack.

Zhi Zhang received the Ph.D. degree from The University of New South Wales. He is a Lecturer at The University of Western Australia. His current research interests include hardware security, system security, and their intersections with AI security. He was a recipient of USENIX SECURITY 2024 Distinguished Paper Award and ASIACCS 2023 Distinguished Paper Award. He serves as an Associate Editor for IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. He also serves on the Program Committees for ASPLOS, DSN, ASIACCS.

Qingni Shen received the Ph.D. degree from the Institute of Software Chinese Academy of Sciences, Beijing, China, in 2006. She is currently a professor with the School of Software and Microelectronics and the director of PKU_OCTA Lab of Blockchain and Privacy Computing, Peking University, Beijing, China. Her research interests include operating system security, cloud security and privacy, and trusted computing. She is a Distinguished Membership of CCF and an ACM/IEEE Membership.

Yansong Gao (Senior Member, IEEE) is a Lecturer at the University of Western Australia. He received his M.Sc degree from the University of Electronic Science and Technology of China and a Ph.D. degree from the University of Adelaide, Australia. His current research interests are AI security and privacy, system security, and hardware security. He serves as an Associate Editor of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Jinhua Cui is an Assistant Professor at the College of Integrated Circuits, Hunan University, China. He received his Ph.D. in Computer Science from the National University of Defense Technology (NUDT) in June 2022. From 2019 to 2021, he was a Research Assistant at the School of Computing, National University of Singapore (NUS). Prior to that, he served as a Senior Research Engineer at the Secure Mobile Centre, Singapore Management University (SMU). Dr. Cui's research lies at the intersection of trusted computing, microarchitectural security, and hardware-software co-design. His work has been published in top-tier venues, including ACM CCS, DAC, DATE, and TCAD.

Yusi Feng received the Ph.D. degree in computer systems organization from Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2025. She is a postdoctoral researcher at the Southern University of Science and Technology. Her research interests include system security and side-channel attacks.

Zhonghai Wu received the PhD degree from Zhejiang University, Hangzhou, China, in 1997. Previously, he worked as a postdoctoral researcher with the Institute of Computer Science and Technology, Peking University, Beijing, China. Since then, he has been involved both in research and development of distributed system, service, and security. He is currently the dean of the School of Software and Microelectronics, Peking University, Beijing, China. His research interests include cloud computing, data intelligence, and system security.

Derek Abbott (M'85—SM'99—F'05) was born in South Kensington, London, U.K. He received the B.Sc. (Hons.) degree in physics from Loughborough University, U.K., in 1982, and the Ph.D. degree in electrical and electronic engineering from the University of Adelaide, Australia, in 1997, under K. Eshraghian and B. R. Davis. His research interests include the areas of multidisciplinary physics and electronic engineering applied to complex systems. His research programs span a number of areas including security, stochastics, game theory, security, photonics, energy policy, biomedical engineering, and computational neuroscience. He is a fellow of the Institute of Physics, U.K., an Honorary Fellow of Engineers Australia and Australian Laureate Fellow. He received a number of awards, including the South Australian Tall Poppy Award for Science, in 2004, an Australian Research Council Future Fellowship, in 2012, the David Dewhurst Medal, in 2015 the Barry Inglis Medal, in 2018, and the M. A. Sargent Medal for eminence in engineering, in 2019. He has served as an Editor and/or a Guest Editor for a number of journals, including the IEEE JOURNAL OF SOLID-STATE CIRCUITS, *Journal of Optics B*, *Chaos*, *Fluctuation and Noise Letters, Royal Society OS*, PROCEEDINGS OF THE IEEE, and the IEEE PHOTONICS JOURNAL.