Electrical-Level Attacks on CPUs, FPGAs, and GPUs: Survey and Implications in the Heterogeneous Era

DINA G. MAHMOUD, EPFL, Switzerland
VINCENT LENDERS, armasuisse, Switzerland
MIRJANA STOJILOVIĆ, EPFL, Switzerland

Given the need for efficient high-performance computing, computer architectures combining central processing units (CPUs), graphics processing units (GPUs), and field-programmable gate arrays (FPGAs) are currently prevalent. However, each of these components suffers from electrical-level security risks. Moving to heterogeneous systems, with the potential of multitenancy, it is essential to understand and investigate how the security vulnerabilities of individual components may affect the system as a whole. In this work, we provide a survey on the electrical-level attacks on CPUs, FPGAs, and GPUs. Additionally, we discuss whether these attacks can extend to heterogeneous systems and highlight open research directions for ensuring the security of heterogeneous computing systems in the future.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Hardware** → *Reconfigurable logic and FPGAs*; • **Computer systems organization** → *Embedded and cyber-physical systems*; *System on a chip*; **Heterogeneous (hybrid) systems**; *Reconfigurable computing*; • **Security and privacy** → *Side-channel analysis and countermeasures*; **Hardware attacks and countermeasures**; *Hardware reverse engineering;*

Additional Key Words and Phrases: Electrical-level attacks, heterogeneous computing systems, CPU, FPGA, GPU

ACM Reference format:
Dina G. Mahmoud, Vincent Lenders, and Mirjana Stojilović. 2022. Electrical-Level Attacks on CPUs, FPGAs, and GPUs: Survey and Implications in the Heterogeneous Era. *ACM Comput. Surv.* 55, 3, Article 58 (February 2022), 40 pages.
https://doi.org/10.1145/3498337

1 INTRODUCTION

A recent trend in high-performance computing is to combine various computing units to achieve better performance. These include central processing units (CPUs), graphics processing units (GPUs), and field-programmable gate arrays (FPGAs). Each of these computing units is better suited for some workloads than others, thus allowing the versatile high-performance computing workloads to exploit the full potential of a *heterogeneous computing system* (HCS).

This research is supported by armasuisse Science and Technology.
Authors' addresses: D. G. Mahmoud and M. Stojilović, EPFL, School of Computer and Communication Sciences, CH-1015 Lausanne, Switzerland; emails: {dina.mahmoud, mirjana.stojilovic}@epfl.ch; V. Lenders, Cyber-Defence Campus, armasuisse Science and Technology, Feuerwerkerstrasse 39, 3602 Thun, Switzerland; email: vincent.lenders@armasuisse.ch.

0360-0300/2022/02-ART58 $15.00
https://doi.org/10.1145/3498337

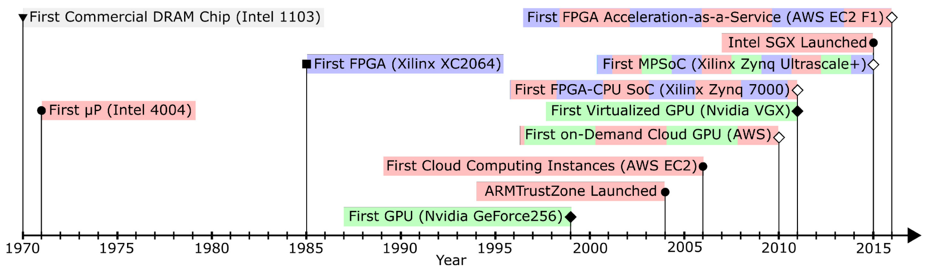

Fig. 1. Timeline of the evolution of computing systems, highlighting the introduction of various compute units, their integration, and their availability to cloud users. CPU milestones are highlighted in red (full circles), GPU milestones are highlighted in green (full rhombi), and FPGA milestones are highlighted in violet (full squares). Heterogeneous milestones are distinguished by the multiple colors (hollow rhombi), while the DRAM milestone is highlighted in grey (full triangle).

In the most generic terms, a heterogeneous system includes a variety of computing platforms with numerous interconnect methods and integration levels between the components. With their chip-level integration, embedded *system-on-chip* (SoC) and *multiprocessor system-on-chip* (MPSoC) platforms are among the most notable examples of HCSs. Besides these plateforms, today's cloud computing architectures are another notable representative of heterogeneous architectures, as they provide the users with a combination of CPUs, GPUs, and FPGAs.

While the concept of heterogeneous computing has been studied for more than two decades [52], with the recent advances in FPGAs and GPUs, the HCSs themselves have greatly evolved. HCSs have become a platform of choice for a multitude of highly parallel tasks, commonly not executed efficiently by CPUs alone, for example, image processing and machine learning (ML) algorithms. The large-scale adoption of heterogeneous systems has, in turn, led to the development of programming standards for hybrid architecture platforms (e.g., OpenCL [201]) and standardization of the architectures (e.g., heterogeneous system architecture (HSA) [90]). Furthermore, as shown in Figure 1, a number of commercial heterogeneous platforms have recently become available, both in the cloud and in embedded form.

However, the security of HCSs is still neither fully investigated nor understood. Not surprisingly, researchers in system security have mostly been focusing on attacks on CPUs; vulnerabilities in CPU software have long been analyzed, starting with the Morris worm in 1988 [51]. Code injection and return-oriented programming are other well-known classes of software attacks [184, 221]. More recently, attacks taking advantage of the processor microarchitecture (e.g., *Spectre* [104] and *Meltdown* [119]) have drawn considerable attention, together with the famous disturbance error-based attack on dynamic random access memories (DRAMs), known as *RowHammer* [102]. The observable side effects of code execution on CPUs—at the *electrical* level, there are power consumption and electromagnetic (EM) emanations and, at other levels, sound waves—have been used for side-channel analysis (SCA) attacks [188]. CPUs are also subject to fault-injection attacks, either via physical access to the device or, as recently shown, remotely [13, 99, 157, 173, 174]. Fault injection can rely on the EM field, voltage, or frequency, but it can also be optical or temperature based [13].

We can observe a similar trend for FPGAs, which have only recently been integrated into a number of cloud infrastructures [6, 7, 91]. FPGAs are known to be vulnerable to fault injection and side-channel attacks when the attacker has physical access to the device [88, 241]. Not long ago, researchers have shown that these attacks on FPGAs can even be controlled and executed remotely [73, 236]. Finally, GPUs, being similar to CPUs in many respects, share a number of their vulnerabilities; microarchitecture-based side channels [96] and power side-channel analysis [124] have been demonstrated on GPUs. Furthermore, FPGAs and GPUs can help accelerate some of the attacks on CPUs— notably, microarchitectural and RowHammer exploits [58, 224].

When CPUs, FPGAs, and GPUs are brought together in a heterogeneous system, the resulting system-level security vulnerabilities are no longer limited to the existing and known exploits of each individual component—new exploits may arise due to the component integration. While not as extensively researched as the attacks focusing on individual computing units, security of HCSs has been looked into. Researchers have demonstrated covert channels, side channels, and RowHammer attacks across the system components [58, 68, 77, 224]. However, more research is necessary to identify, understand, and overcome the security risks brought up or aggravated by the close integration of diverse processing units.

In this article, we present a survey of the attacks on CPUs, GPUs, and FPGAs. In particular, we focus on electrical-level attacks because, unlike microarchitectural or software-based attacks, their underlying mechanism is typically not platform-dependent. Hence, they have the potential to affect all considered processing units. We examine RowHammer, fault injection, and SCA attacks. Unlike other surveys, which focus on only one compute unit, one exploit type, or one threat model [15, 158, 179, 188, 211, 232, 233, 237], we include in our examination all three computing units and take into consideration the heterogeneous system architecture and deployment models.

The rest of this article is organized as follows. We start by presenting the heterogeneous system architectures and the associated threat models in Section 2. We then provide a classification of the attacks in Section 3. We survey and discuss the known attacks on CPUs, FPGAs, and GPUs in Sections 4, 5, and 6, respectively. In Section 7, we discuss whether the electrical-level attacks described previously can affect heterogeneous systems and which new attacks may be feasible in a heterogeneous setting. We present our conclusions in Section 8.

2 HETEROGENEOUS SYSTEMS AND THREAT MODELS

We begin a thorough analysis of the electrical-level vulnerabilities of heterogeneous systems by examining their common architectures and variations. Widely deployed embedded heterogeneous platforms include the SoCs by Xilinx and Intel, which tightly integrate CPUs with FPGAs [8]. Xilinx even offers an MPSoC option with GPUs [242]. In the cloud market, both Intel and Xilinx have their share, with the accelerator cards deployed in AWS [7], Microsoft [142], Alibaba [6], and Huawei [91] cloud instances. Of course, GPUs are available as well. In this section, we distinguish heterogeneous systems by their integration style: *chip level* and *system level*. We show a generic heterogeneous architecture in Figure 2 and highlight the differences between the two integration styles in the following subsections. We end this section with the explanation of the threat models.

2.1 Chip-Level Heterogeneity

Chip-level heterogeneity is heterogeneous integration inside the device package. As a generic example, we choose here the Zynq UltraScale+ MPSoC from Xilinx [242], which combines an ARM processor with a GPU and an FPGA. It has an application processing unit—the CPU of the system—and a real-time processing unit. The components communicate via the advanced extensible interface (AXI) [242] (*communication buses* in Figure 2), common in Intel SoC platforms as well (e.g., Cyclone V [45] and Arria 10 [8]).

Access to the main memory is often shared among components; on heterogeneous CPU-GPU platforms, both components can access the main memory, and the GPU can use both coherent and noncoherent communication [32]. In the Xilinx UltraScale+ MPSoC device, all three components

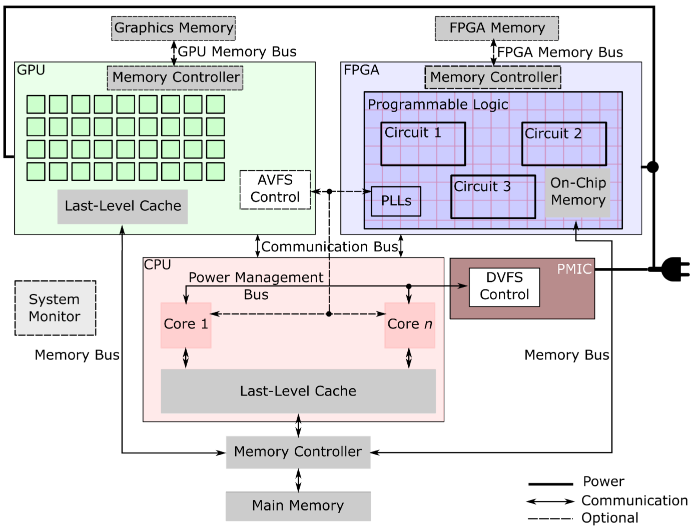

Fig. 2. A representative example of a heterogeneous system architecture. Optional parts are shown with dashed lines. The connections between components serve for communication or power delivery (thick lines).

can access the external DRAM through the memory controller, which exposes the system to the risk of RowHammer exploits. Additionally, the memory can be partitioned to provide isolation between multiple users or applications [140], which may provide some protection. Memory-to-memory and memory-to-IO transfers take advantage of the direct memory access (DMA)—common in Intel Cyclone V and Arria 10 SoCs as well [8, 45]—with the number and rate of access requests handled by the transaction control mechanism [242].

The ARM processor supports the trusted execution environment (TEE) called TrustZone, a target of recent attacks, and frequency scaling, a facilitator of recent attacks [8, 45, 242]. Software interfaces allow the CPU cores to communicate the desired frequency to the dynamic voltage and frequency scaling (DVFS) control via the power management bus (PMBus). Sometimes, GPUs allow frequency scaling (controlled from the CPU side [63], and known as adaptive voltage and frequency scaling (AVFS)), but that is not the case in the UltraScale + MPSoC. For FPGAs in general, the clock management resources—phase-locked loops (PLLs) and mixed-mode clock manager modules—can be configured by the FPGA applications or by the CPU. While FPGA voltage scaling is typically not supported, if the on-board voltage regulators support it, the PMBus may be used to control the voltage supplied to the FPGA [186]. Since the programmable logic (PL) shares the power distribution network (PDN; the FPGA grid in Figure 2), the result is free propagation of voltage disturbances across the die and, consequently, a number of electrical-level attacks on FPGAs (Section 5).

In the SoCs with an ARM processor (e.g., Intel Cyclone V), it is possible to poll the CPU power and temperature from software [45]. On the other side, Xilinx UltraScale+ MPSoC is equipped with a system monitor, which reports the voltage and temperature at various on-chip locations (in the processing system and the PL). This MPSoC is divided into power-gated islands for turning off unused parts of the system—a feature not unique to Xilinx SoCs [45]. The existence of power-gated islands could suggest that different system components require different voltage levels; in reality, this is often not the case. As a consequence, the PDN design is simplified (the number of voltage regulators reduced), but the risk of electrical-level attacks is increased (as discussed in Section 7).

2.2 System-Level Heterogeneity

Datacenters (and the cloud) are a typical example of system-level heterogeneity. In datacenters, GPUs and FPGAs lie on their respective accelerator cards, plugged into the server racks and connected with the CPUs through peripheral component interconnect express (PCIe), quick path interconnect (QPI), or ultrapath interconnect (UPI). The last two are common to Intel Xeon + FPGA platforms, available to researchers through the Intel Hardware Accelerator Research Program (HARP) [38]. Even though the coupling through the common power supply is lower than for chip-level integration, it is not fully eliminated, because all of the cards in one rack share a common power supply unit.

Besides having access to the CPU's main memory, FPGAs and GPUs can also have their own memory modules (Figure 2). In that case, for performance reasons, data are typically moved in bulk between the CPU main memory and the accelerator memory [38]. The specifics of voltage and frequency scaling largely depend on the CPU and GPU used. As far as FPGAs are concerned, the clock management resources can be configured by the users. Access to voltage and frequency scaling interfaces can sometimes be hindered by virtualization: CPUs are typically virtualized, while GPUs can be virtualized, for example, using rCUDA [48] or the NVIDIA virtual GPU [217]. While many virtualization proposals exist for FPGAs [176], the hardware of cloud FPGAs is currently not abstracted away. Finally, despite the number of research works on voltage and frequency scaling on FPGAs [4], these techniques are not yet supported in heterogeneous settings.

2.3 Threat Models

Regardless of the integration level, various assumptions can be made on how an adversary may have access to the platform and what the adversary can do depending on the deployment scenario. From appliances to Internet-of-Things (IoT) devices, embedded systems are everywhere. Many of the embedded devices (e.g., IoT sensor nodes) are connected to the network for sending their data to the users or providing access to the device itself. In this case, the heterogeneity is typically at the chip level. Even though multitenancy is typically not considered in embedded scenarios, hardware Trojan horses from third-party intellectual property or malicious chip manufacturers, as well as malicious software, can create covert communication channels or interfere with the operation of the device. Both an attacker with direct access to the hardware (with the possibility of removing the integrated circuit (IC) package or probing and measuring the external signals) as well as an attacker with remote access are valid threat model assumptions. Furthermore, both chip-level and system-level heterogeneous devices may be the attack targets.

In the cloud deployment scenario, a malicious user with remote access may be assumed to either have user privileges or to have gained (legally or not) root privileges. For FPGAs, a typical assumption is that each device has a soft programmable shell that manages the use of FPGA resources. At the same time, the FPGA users can share the logic and hardened resources temporally or spatially. In the case of spatial cotenancy, the roles are assumed to be logically separated, while the PDN remains common. It is interesting to note that, even though there are a number of proposed solutions for FPGA virtualization [214] (driven by the benefits of multitenancy and low implementation effort, thanks to the partial reconfiguration feature of FPGAs), due to the recently discovered electrical-level security risks that FPGA multitenancy creates, no cloud service provider (CSP) allows it yet.

If an adversary has physical access to the datacenter rack or root privileges, then some threat models assume the availability of TEEs-for example, ARM TrustZone or Intel software guard extensions (SGX)—for isolating the execution of users' code. However, TEEs do not provide full protection against all adversaries with physical access [37]. In the case of FPGAs, physical access means control over the shell and, consequently, over the clock or data.

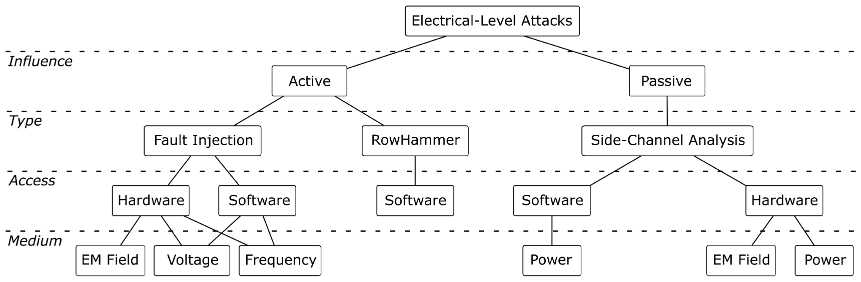

Fig. 3. Classification of the electrical-level attacks common to CPUs, FPGAs, and GPUs.

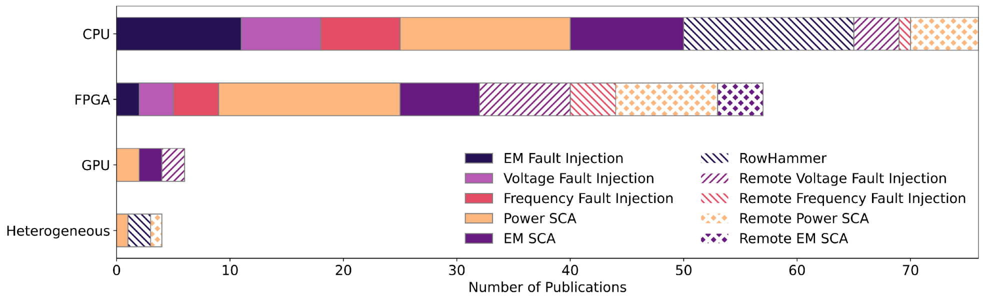

Fig. 4. Current state of research on the electrical-level attacks targeting individual processing units or heterogeneous systems.

3 BACKGROUND AND TAXONOMY

Electrical-level attacks are common to all system constituents, albeit the exact interface exploited to carry them out may differ depending on the target. These attacks can have an *active* or a *passive* influence on the target. A passive attack simply observes the target, without affecting it. An active exploit, however, requires interfering with the target's functionality. Figure 3 illustrates the taxonomy of electrical-level attacks common to CPUs, FPGAs, and GPUs. Figure 4 highlights the current state of research (approximated by the number of publications addressing new or enhanced attacks that we cite in this article) and, in a way, applicability of a particular type of attack to either a specific processing unit or a heterogeneous system. The figure shows that CPUs and FPGAs have so far been the most attractive attack targets. In the following subsections, we introduce the three main attack types and further explain the taxonomy.

3.1 RowHammer

We consider *RowHammer* as a distinct type of active software-based electrical-level attacks. The goal of RowHammer attacks is to induce errors in DRAM, with consequences affecting the entire system.

DRAM process technology is continuously scaled to pack a larger number of smaller cells in one module. However, this comes at a cost of reduced memory reliability, as the smaller cell size and the tighter spacing between the cells favor EM coupling effects, which can affect the leakage rate of the charge stored in each cell, representing one bit. Other scaling effects include the reduced noise margins and higher variation in process technology [102].

If a cell leaks more charge than its noise margin, a fault occurs [102]. To avoid the leakage, DRAM rows of cells are regularly refreshed. Nonetheless, attackers can leverage a phenomenon known as *disturbance* to exploit the leakage in DRAM cells and induce faults. This phenomenon occurs when neighboring cells interfere with each other's operation; when a cell is disturbed beyond its noise margin, it malfunctions, resulting in a *disturbance error*. In 2014, Kim et al. showed that repeatedly accessing a row in the DRAM can interfere with neighboring cells, resulting in disturbance errors [102]. DRAM manufacturers employ circuit-level design techniques (to improve the isolation between cells) and post-production testing to ensure that chips arriving to the users do not suffer from disturbance errors. However, with the continued scaling of DRAM cells, these mitigations are no longer sufficient. Kim et al. highlight this insufficiency by showing that DRAM modules manufactured after a specific date are prone to RowHammer-induced faults [102].

Inducing bitflips in the target DRAM requires the adversary to be able to access it. This access is possible when the adversary resides on the same system or can use DMA over the network (e.g., Throwhammer [207]). The feasibility of inducing RowHammer bitflips mainly depends on the target DRAM, but researchers keep demonstrating the feasibility even on DRAM modules previously assumed to be immune [59]. As will be discussed later, a successful attack also requires the bitflips to be induced in an exploitable location, which can be ensured through some knowledge of the target, physical addresses, or the electrical or microarchitectural side channels. Bitflips in other locations may either crash the system or not prove useful to the attacker. Since the adversary does not need to access the victim row, access to the part of the memory belonging to the victim is not required. However, in some cases, memory deduplication is useful for enabling the attack by sharing pages between the victim and the adversary [180]. The RowHammer attack vector is particularly dangerous because it is entirely software controlled, not requiring access to the hardware. Moreover, RowHammer has been used in realistic scenarios, where the CPU was the intended victim. Most of the exploits were initiated from (and targeting) the CPU, but a few were launched from an FPGA, a GPU, or the network [58, 207, 224].

3.2 Side-Channel Analysis

For decades, it has been known that computing systems leak information through side channels (e.g., power consumption or electromagnetic emanations) [208]. This leakage has been exploited as an attack vector for reverse engineering [39, 175] and for breaking the security of cryptographic primitives [105].

Electrical side channels are created by the existence of mutual information between the code being executed and the data being processed on the one side and the resulting power consumption and EM emanations on the other side. The most common techniques for extracting the information are simple power analysis [179], differential power analysis (DPA) [105], correlation power analysis (CPA) [29], template attacks [36], mutual information analysis [70], and machine learning [84]. Machine learning approaches are particularly adapted for side-channel disassembly (reverse-engineering of the executed code, mainly on a CPU), as the dimensionality of the problem and the size of the data to be recovered is much higher than in the case of side-channel cryptanalysis [166].

SCA can be seen as a passive attack, as the main requirement is the observation of the side effects of the code execution. The side-channel information may be obtained through hardware access or through a software interface (recent research trend for FPGAs). When an adversary has access to the hardware, the SCA may be *invasive* if, for instance, the IC package needs to be removed or manipulated. Most commonly, it is *noninvasive*. In a laboratory, researchers may opt for the invasive model, adding resistors for measuring the power consumption or decapsulating the IC for observing EM emanations. In realistic settings, depending on where the victim is deployed and how it is protected, a noninvasive attack may be more appropriate (leaving no trace of the attack). If the adversary resides on the same system as the victim, then software-based exploits are feasible through power monitoring interfaces or the ability to reconfigure FPGA logic to act as a sensor. If the victim provides a public interface through which the adversary may send input data, then the attacker can fix the input to a predefined value and collect leakage traces to learn the secret (i.e., known plaintext attack). EM-based side channel exploits (except for crosstalk coupling on FPGAs) require hardware access to the victim. However, unlike hardware power-based SCA, EM-based SCA does not require adding a resistor for measurements or connecting a voltage probe to one of the IC's pins. Rather, it requires only an EM probe positioned near the IC.

3.3 Fault Injection

The clock frequency and power supply voltage govern the correct operation of digital circuits, including the ones inside a processor. For a specific frequency value, there exists a minimum voltage value, known as the critical supply voltage, to ensure correct operation of the circuits. If these values are not respected, the circuit experiences errors.

In 1997, Boneh et al. introduced the threat of fault injection to cryptographic circuits, highlighting how to use hardware faults, leading to random transient bitflips, to break Rivest–Shamir–Adleman (RSA) and Rabin signatures [23]. Such transient faults can be injected in many ways; if the clock frequency is too high or the clock signal contains glitches, inputs to registers in the circuit may not arrive early enough to be processed by the register, resulting in early latching or metastability. Even if the clock frequency is within limits, lowering the supply voltage results in longer signal delays, which again can lead to computational faults. The number and distribution of faults depends on whether the voltage has fast (glitches) or steady changes [97]. Low-power applications pushed for the emergence of DVFS techniques, which, as it turns out, can be abused for remote fault-injection attacks against CPUs and GPUs [173, 185, 206].

Fault-injection exploits are active attacks, because they require interfering with the target operation. They can be carried out via software access only (e.g., by controlling the DVFS interfaces of the CPU or by programming the FPGA). If available, hardware access may give the adversary direct control over the power supply or the clock, thus simplifying the fault injection. In a lab setting, to facilitate the attack, the on-board capacitance could be manipulated or the chip connected to the attacker-controlled power supply [209]. Even though such changes may not be possible in all realistic scenarios, they allow researchers to assess the susceptibility of the victim to the exploit. Attack success can sometimes be facilitated by prior observations of side-channel leakage to determine when precisely to inject the faults. Exploiting the injected faults typically requires the attacker to be able to observe the output (e.g., for differential fault analysis) or run a software process on the victim device to gain special privileges thanks to the injected faults.

4 ELECTRICAL-LEVEL ATTACKS ON CPU-BASED SYSTEMS

Let us begin the survey of attacks (and the corresponding defenses) focusing on the CPU-based systems. Figure 5 shows their simplified architecture and highlights both the components and interfaces vulnerable to fault and side-channel attacks. In the following subsections, we discuss them all, following the taxonomy introduced in the previous section.

4.1 RowHammer Attacks

Kim et al. were the first to exploit the DRAM memory sensitivity to disturbance faults for creating bitflips [102]. In what they named a RowHammer attack, ① in Figure 5, disturbance errors were induced by repetitive opening of a DRAM row, access (read or write), and closing.

Depending on the location of the bitflip, various effects can occur: for example, bypassing the authentication carried out by cryptographic codes, application crashing, persistent loss of data, or file system damage [207]. Modern DRAMs, with their increased density and without mitigations, are

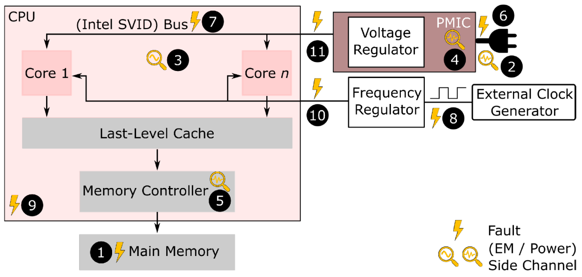

Fig. 5. Illustrative system architecture for a CPU-only system, with interfaces vulnerable to fault and side-channel attacks highlighted. The circled numbers indicate the vulnerabilities we discuss. The symbols indicate the attack type.

more vulnerable than older DRAMs to RowHammer [101]. On recent DDR4 modules, which were previously considered immune to RowHammer, Frigo et al. demonstrated a *many-sided* RowHammer attack (with as many as 19 aggressor rows). They uncovered how to bypass the target row refresh countermeasure, deployed in DDR4 [59].

RowHammer attacks can be combined with cache or memory deduplication side channels to cause more harm to the system. Seaborn and Dullien were the first to achieve privilege escalation from a process running locally on the victim machine by flipping bits in the physical page number of page table entries [192]. Privilege escalation from an unprivileged Android app was also demonstrated on ARM mobile devices by van der Veen et al., who relied on the predictability of memory allocation patterns to ensure that the victim will allocate its page table entries in RowHammer-vulnerable memory locations [215]. TEEs are also affected by RowHammer bitflips, which can be leveraged by a remote attacker with no system privileges for denial-of-service (DoS) attacks on SGX enclaves [94].

While the requirement for main memory access suggests the need for a flush instruction (to bypass the cache), recent works have demonstrated successful RowHammer exploits without it: Qiao and Seaborn used x86 nontemporal instructions to bypass the cache [171], Aweke et al. used eviction sets [10], useful for remote exploits by unprivileged adversaries as well [80], while Van der Veen et al. used DMA on ARM mobile devices [216].

Coupling the RowHammer attack with a website with JavaScript makes for a remotely controlled attack, allowing the adversary to gain memory access, or break the memory isolation and access another virtual machine on the same CPU [25, 80, 180]. An adversary can also use network packets to induce bitflips in the memory [120], and remote DMA can allow a foe to gain code execution rights on a key-value server application [207]. Most recently, Cojocar et al. formalized the method for determining whether a cloud server is vulnerable to the RowHammer attack and designed a fault injector that enables reverse engineering the row adjacency map of DDR4 modules [41].

4.1.1 Countermeasures. Given that decreasing the refresh interval of DRAM modules can impede and, at a certain point, stop RowHammer attacks, the first proposed countermeasure was to increase the refresh rate. This was implemented by some of the memory manufacturers, for example, Apple and HP [158]. However, the refresh rate increase comes at the cost of higher energy consumption, lower system performance, and a degradation in the quality of service. Kim et al. found that a refresh rate increase by a factor of at least 7.8× was needed to eliminate all RowHammer-induced errors [102]. Other countermeasures can be classified as software- or hardware-based.

ANVIL is an example of a software-based countermeasure, which makes use of the hardware performance counters to determine whether a memory access pattern indicative of a RowHammer attack is occurring and, if needed, refreshes the affected DRAM cells [10]. Alternatively, one can modify the page allocation to ensure that an attacker cannot impact important memory pages [28]. GuardIon takes a similar approach, isolating the DMA buffers on ARM devices to prevent DMA-based RowHammer [216]. ZebRAM does this as well—it isolates every row containing data with guard rows, which can be filled with other data and protected using integrity checks [106]. Vig et al. proposed using a dynamic skewed hash tree to quickly detect RowHammer exploits [218]. It has been found that static code analysis can help identify malicious software (e.g., MASCAT [93]). However, all software-based solutions are costly, as they require modifications to the system software, incur performance or memory overheads, or interfere with system operation [158].

Turning to the hardware-based solutions, some are quick to implement—for example, remapping or retiring vulnerable cells in the DRAM. While not requiring significant hardware changes, discarding the unsafe cells requires first their dynamic identification, resulting in time, effort, and storage overhead [102]. Other short-term options make use of the error-correcting code modules, if available. However, error-correcting codes incur considerable overhead and often correct only single-bit errors, while RowHammer can induce multiple bit flips [102]. On the other hand, some hardware solutions focus on long-term protection. Modifications to the memory controller or to the memory module—such as the probabilistic adjacent row activation (PARA) mechanism proposed by Kim et al. [102] and the probabilistically managed table by Son et al. [198]—provide good security guarantees. Some versions of probabilistic defenses are already deployed in recent Intel memory controllers [158]. Other possibilities include adding hardware counters (e.g., counter-based tree [196] and time window counters [116]) in the memory controller. More recently, Yağlıkčı et al. proposed BlockHammer, which throttles the rate at which a single DRAM row can be accessed to avoid bitflips [227]. BlockHammer also limits memory requests by suspicious threads, ensuring that benign threads can still benefit from a good memory bandwidth. EM side channels, which we discuss in the following subsection, have also been proposed as detection methods for RowHammer attacks [235]. While many countermeasures are discussed, the deployment of effective defenses is still underway, as new research demonstrates persisting vulnerabilities.

4.2 Side-Channel Attacks

CPUs have long been a target of side-channel attacks [105], with the goal of breaking the security of cryptographic algorithms or, more recently, reverse-engineering the code running on the CPU (i.e., side-channel disassembly). SCA attacks typically require hardware access to the device, but recent works have shown the feasibility of software-based exploits via power management interfaces [118]. Here, we discuss both hardware- and software-based exploits.

4.2.1 Hardware Side-Channel Attacks. An adversary with physical access to a device can gain information about the code executing on that device, often sufficient for compromising secure data and proprietary code. In what follows, we examine the use of that information for side-channel cryptanalysis and disassembly. While we focus on the malicious exploitability of side-channel disassembly, it is worth noting that it can also be used for system protection [83, 165].

Power. From the first efforts to develop a power-based SCA [105], ② in Figure 5, the hardware security research community has shown significant interest in these exploits. Since the works showing the dangers of variable latency of cryptographic operations [79], algorithms have improved to overcome obvious vulnerabilities. Yet, researchers are continuously bringing to light new exploits [179, 237]. Genkin et al. underscored the extent of the threat of power SCA by extracting 4096-bit RSA keys and 3072-bit ElGamal keys from personal laptops, using power and EM measurements at the chassis and at the end of a USB or an HDMI cable [65]. Surely, design choices play an important role in side-channel resistance: Biryukov et al. analyzed the factors that render a cryptographic implementation more secure against a CPA attack and validated their findings on eight different lightweight ciphers and an 8-bit ATmega2561 processor [22].

A successful side-channel attack typically requires multiple measurements (i.e., traces) [40]. To overcome this limitation, Clavier et al. proposed horizontal correlation analysis, which, with one known message to encrypt, exploits the similarity of operations and data within one algorithm execution [40]. They used a single power trace from an exponentiation process of the RSA algorithm on a RISC processor. Even though their method does not work against all implementations (e.g., RSA with Chinese remainder theorem), it demonstrates that there are approaches that help optimize the exploit. Batina et al. used a single trace to learn the input to a publicly available multilayer perceptron (MLP), relying on the sequential computations in an ARM Cortex-M3 [18]. Given that in the first layer of a neural network (NN) multiple weights (known in this scenario) are all multiplied by the same input (the secret), the side-channel traces during multiplication enable the recovery of the secret. While ML models can be targets of exploits, they can also help facilitate the attacks [85, 89] or decrease the number of required traces [223]. Masure et al. proved that the optimization that takes place during ML training is equivalent to maximizing the obtained side-channel information [136].

Parallel to works using side-channel leakage to reveal secret information processed by a CPU, researchers began to investigate whether this leakage could disclose more secrets, notably the executed instructions. On a Zilog Z80 processor, Quisquater and Samyde were the first to demonstrate automated instruction reverse-engineering using power and EM traces [175]. Their use of a Kohonen NN was indicative of the observed trend of employing ML in subsequent research on side-channel disassembly. Eisenbarth et al. instead relied on hidden Markov models to build a power side-channel disassembler targeting a PIC microcontroller and resulting in a recognition rate of up to 70% [50].

Recent works continue the trend of ML-assisted disassembly [113, 166, 212]. Park et al. focused on attacks on IoT devices, conducting experiments on the AVR ATmega328p processor and analyzing 112 instructions and 64 source and destination registers [166]. Taking advantage of continuous wavelet transform to observe differences between instructions not observable in the time domain, Park et al. then applied a feature selection method to reduce the required number of samples. They examined a number of classifiers: linear discriminant analysis, quadratic discriminant analysis, naïve Bayes, and least squares support vector machine (LS-SVM) and developed a full disassembler for monitoring the code execution on embedded processors, with security as the main aim. They used hierarchical classification, thus reducing the computational complexity, and a covariate shift adaptation to overcome the effects of changes in the measurement environment. With quadratic discriminant analysis, the classification accuracy reached 99.03% [166].

When the number of clock cycles per instruction (CCPI) varies, disassembly becomes more complicated. Krishnankutty et al. found the optimal sequence of clock-cycle lengths on a general-purpose pipelined processor (MSP430G2553) using empirically created templates together with dynamic programming [113]. Once the CCPI is known, the next step is to find the correct instruction among those with the given CCPI. Between the two steps, one can resort to a coarse-grained classification according to hardware usage, which Krishnankutty et al. implemented using an SVM (92%–100% accuracy). In comparison, the fine-grained classifier tested on real instruction sequences (advanced encryption standard (AES) and proportional-integral-derivative controller) resulted in about 86% accuracy. Similarly, Vafa et al. used a hierarchical framework for instruction recognition on an 8-bit PIC and a 32-bit ARM Cortex-M3 microcontroller, with both static and dynamic power measurements, achieving an 80% to 93% recognition rate [212].

EM Field. Power traces are not the only side channel, as EM emanations, ③ in Figure 5, can provide even more information about the code execution on different parts of a CPU [175]. Similar to their power SCA attack on laptops, Genkin et al. used EM traces to extract encryption keys [64]. Sehatbakhsh et al. found the number of characters typed and even entire passwords typed on computers [193]. SoCs are also at risk; Longo et al. focused on the AM335x SoC's CPU, cryptographic coprocessor, and the NEON single instruction multiple data (SIMD) coprocessor and addressed the challenges of collecting CPU traces while accounting for interrupts and synchronization issues [123]. Their analysis highlights the difference in leakage among the SoC components while demonstrating the feasibility of the attack.

Understanding and quantifying information leakage can enable more effective attacks and facilitate the quest for countermeasures [229]. In their work on side-channel-aware software engineering, McCann et al. described a technique for modelling data-dependent leakage from instructions [139]. The knowledge of the sources of leakage permitted Primas et al. to break the security of a lattice-based encryption scheme running on an ARM Cortex-M4F CPU, using only a single trace obtained by combining the leakage of the entire decryption process [168].

Neural networks are also not immune to EM-based SCA. Assuming an attacker who can send inputs to the target device, Batina et al. used correlation analysis to recover the weights of an NN, along with a combination of simple and differential analysis to find the number of its layers and weights. Their targets were a decapsulated ATmega328P processor and an ARM Cortex-M3. For inferring the activation function, however, they relied on a timing side channel [17]. Soon after, Takatoi et al. showed how to find the activation function using simple EM analysis [205]. Robyns et al. used convolutional neural networks (CNNs) for another purpose—to classify EM traces corresponding to various encryption instructions of a NodeMCU device [183]. In SCANDALee, Strobel et al. chose to attack PIC16F687, in which most of the instructions take four clock cycles to be executed, making the task of identifying the boundaries between them significantly easier [204]. Polychotomous linear discriminant analysis was used for preprocessing to emphasize the particularly distinctive features in the observed data, classified by a k-nearest-neighbors classifier. The authors reported recognition rates of 96% on test data and 88% on real code. However, their measurement setup required decapsulation of the target device, leaving physical traces of the attack. Decapsulation is not always a requirement; e.g., Vaidyan and Tyagi collected the EM emanations from an Atmega328 processor without decapsulating the IC [213].

4.2.2 Software-Based Side-Channel Attacks. For power efficiency, modern CPUs typically provide users with software access to power consumption information and control of the associated parameters. This leads to software-based side-channel exploits, ④ in Figure 5. Yan et al. collected power information from the battery management unit (BMU) on smartphones, allowing application identification, user interface inference, password length guessing, and geolocation estimation, all of which constitute user security breaches [225]. Qin and Yue proposed a power estimation method based on SVMs and requiring no root privileges. They used it for website fingerprinting on Android 7 [172].

It is important to note that side-channel security is not guaranteed by TEEs: O'Flynn and Dewar successfully leveraged the on-board analog-to-digital converter on a Cortex-M processor, with TrustZone, for remote SCA across security domains [164]. Intel devices are similarly vulnerable: By reading the Intel running average power limit (RAPL), Mantel et al. could distinguish between different RSA keys [134]. Lipp et al. showed the feasibility of power SCA on x86-powered laptops and servers based on unprivileged access to the RAPL [118]. Their *PLATYPUS* exploits compromise the security of Intel SGX enclaves, as an unprivileged adversary can leak AES-NI keys while a privileged one can recover RSA keys. Even the less expected interfaces can facilitate exploits; Gravellier et al. demonstrated the risks associated with allowing software access to delay-line components on SoCs [76]. They used delay-locked-loops and programmable delay-blocks, implemented in memory controllers for the alignment of the sampling clock and the data signals, to carry out a CPA attack, ⑤ in Figure 5, against OpenSSL AES-128 on two SoCs (both with ARM processors).

4.2.3 Countermeasures. An effective software countermeasure involves automated code modification. However, it comes at a cost of decreased performance and increased memory requirements [19, 219]. Alternatives are lightweight cryptographic primitives, more efficiently masked [78, 167], or random code injection for obfuscation [225].

Decreasing leakage from the hardware itself typically requires its redesign. De Mulder et al. integrated defenses into the hardware architecture of a soft-core RISC-V processor and validated them on a Zynq FPGA. To reduce leakage, they implemented masking in the CPU and added protection for the memory accesses [156]. PARAM is another instance of a side-channel hardened microprocessor, implemented and evaluated on a SASEBO-GIII FPGA [56]. The authors first analyzed the RTL of a RISC-V processor to identify leaky modules and then applied obfuscation to reduce leakage in the datapath and hide the addresses sent to the cache. Finally, restricting access to the power-reporting interfaces or limiting their measurement resolution (time or energy) can hinder software-based exploits [118].

4.3 Fault Injection

Instead of monitoring the emanations from a CPU, an attacker with hardware access can inject errors in the operation of the CPU. Moreover, recent efforts to reverse-engineer the DVFS interfaces on modern processors opened the way for fault injection via software access only. In what follows, we present exploits with both access models (see Figure 3).

4.3.1 Hardware-Based Fault Injection. An attacker with physical access to the victim can change the voltage supply or the clock frequency, or inject glitches. The goal may be to fault CPU operations or crash the software, for example, through a fault injected into the opcodes of the executed instructions. Alternatively, the attacker may want to gain information from the results of the faulty computations, in which case access to the victim's output is typically required.

Voltage. Manipulating the supply voltage to affect the operation of processors, ⑥ in Figure 5, has long been investigated. Barenghi et al. studied the effects of undervolting an ARM9 CPU with a 5-stage pipeline [14]. They found that only the `LOAD` instructions were affected due to their long datapaths, resulting in two possible errors: instruction swapping and errors in data loading. In the first case, the fault occurs during the instruction load or the evaluation of a branch condition, resulting in a wrong instruction being loaded. In the second case, the loaded data are faulty. Barenghi et al. used undervolting against AES, knowing that the faults would be injected in the values loaded from memory and leveraging them for a differential fault analysis (DFA) attack [14]. Additionally, they attacked the RSA signature and encryption primitives, managing to recover the private key through a single faulty signature and enable decrypting of an RSA message when in possession of one faulty and one correct encryption of the same unknown message [14].

Carefully choosing the fault injection parameters—that is, the glitch amplitude, duration, and number of occurrences—can make the difference between a successful and failed exploit. Carpi et al. tested various strategies for finding successful attack parameters on smart cards [35]. Bozzato et al. proposed a new voltage fault-injection setup to generate arbitrary voltage glitches [27] and used it to extract the firmware of six different microcontrollers by attacking the commands that give read access to the adversary.

Voltage-based fault injection can be exploited for other purposes as well. Timmers and Mune injected glitches into the power supply line to fault the U-Boot of an ARM processor and load attacker-controlled values into the program counter register [209]. With glitch-injected faults, they also managed to access physical memory addresses and escalate privilege levels. Spruyt et al. proposed using fault injection in a manner similar to SCA. They injected faults into the victim circuit and turned the observed faults into probability traces, which are correlated with the power consumption [199]. Cojocar et al. showed that some of the common countermeasures—for example, instruction duplication and *n*-plication—are not effective against fault injection attacks on ARM Cortex-M4 processors [42].

Most recently, hardware fault injection was used against a “full-fledged Intel CPU”: Chen et al. reverse-engineered how the voltage level is controlled on Intel processors [37]. In their so-called *Voltpillager* attack (⑦ in Figure 5), the authors used the serial voltage identification (SVID) bus, on which they connected a microcontroller-based system, to send the required voltage value to the voltage regulator. Chen et al. demonstrated some faults in the elementary operations (e.g., delaying writes from the memory to the cache) and showed the same proof-of-concept attacks as the remote *Plundervolt* attack (discussed later in this section). Intel’s current mitigations against Plundervolt are ineffective against Voltpillager, as the latter has a stronger access model. TEEs are not effective either, as they do not provide tight-enough security guarantees in such a threat model [37].

Frequency. Fault injection is possible by changing the frequency of the processor’s clock, ⑧ in Figure 5, or by injecting clock glitches. Bar-El et al. showed that clock glitches can be leveraged to skip instructions and, hence, to execute a subset of program instructions, as well as to corrupt the processed data [13]. Korak and Hoefler studied the effects of clock glitches on an ARM Cortex-M0 and an ATxmega 256, and proposed the combination of voltage and clock glitches to improve the reproducibility of the injected faults [107]. For a higher success rate and to get a larger time window for the fault injection, Korak et al. combined heating with clock glitches [108]. Injected faults are not always easy to exploit; looking at 8-bit microcontrollers, Balasch et al. highlighted the complexity of exploiting injected faults because they may affect two stages of the pipeline at the same time, whereas which stages are affected depends on the parameters of the clock glitch [12].

The design of a glitch injector is essential for the attack success. Kazemi et al. proposed two clock glitch generators, both implemented on an FPGA, and used them to fault an ARM Cortex-M3 microcontroller running an AES encryption [98]. Obermaier et al. proposed a *fuzzy glitch* generator based on ring oscillators (ROs) and used it to inject faults into an ARM Cortex-M0 microcontroller [160]. Using PLLs to generate the clock signal from an external clock source does not provide enough protection against clock glitches; Selmke et al. managed to force the PLL to overclock using high-frequency signals [195]. Since the PLL generates the internal clock signal by modifying the input clock signal (multiplying or dividing the frequency by a factor), changing the frequency of the external clock for enough clock cycles will result in the PLL also adjusting its output clock signal.

EM Field. Instead of manipulating the voltage supply to inject faults, an attacker can channel high-amplitude voltage pulses through an EM probe, ⑨ in Figure 5. The generated EM pulses can lead to timing constraint violations or (re)setting of flip-flops [161]. Manipulating the IC itself can affect the fault injection; Schmidt and Hutter compared the errors induced in a microcontroller when it is encapsulated, front-side decapsulated, and back-side decapsulated [191]. For a nearby probe, front-side decapsulation resulted in the highest injected current. For a probe further away, no difference among the three scenarios was observed. Dehbaoui et al. confirmed that EM fault injection can work without tampering with the victim device by carrying out a DFA attack against an AVR microcontroller running AES encryption [46].

Moro et al. built a basic fault model for an ARM-Cortex M3, while investigating the effects of the probe's position and the timing of the pulses [151]. Trouchkine et al. used EM pulses to characterize fault-injection effects on a BCM2837 (ARM) and an Intel Core i3 to determine which type of fault occurs (memory corruption, register corruption, or bad fetch) and which microarchitectural block is affected (the registers, cache, pipeline, memory management unit, or the bus) [210]. Maldini et al. went one step further and optimized the parameters of the exploit with an evolutionary algorithm [132]. Menu et al. exploited the localized effects of the EM-fault injection (against data transfers from the flash memory to the data buffer of a Cortex-M microcontroller) to (re)set key bytes, carry out a DFA attack, and inject faults that persist until reboot, ultimately recovering the secret AES key [141].

EM fault injection can be used for a variety of attacks: Rivière et al. targeted the instruction cache of an ARMv7-M architecture to steer its control flow, achieving a high reproducibility rate [182]. Elmohr et al. demonstrated multiple instruction faults and skips on a RISC-V processor and an ARM Cortex M0 [53]. Cui and Housley proposed leveraging second-order effects of EM fault injection, in which a fault in one component can affect other components, to overcome the inability to use first-order effects (EM field injects faults into a specific component due to proximity) in modern processors, which run at frequencies higher than 1 GHz [44]. Their approach caused software execution to enter into a data-dependent fault condition that is normally unreachable on a TrustZone-equipped Internet protocol phone. Dehbaoui et al. employed EM-induced faults against the round counter of an AES implementation on an ARM Cortex-M3 processor, to induce another execution of the penultimate round. This allowed for a DFA attack with only two pairs of correct and faulty ciphertexts [47].

4.3.2 Software-Based Fault Injection. Until recently, manipulating the voltage or frequency supplied to a processor required having physical access to the device. This no longer holds, because researchers have reverse-engineered and discovered how to use DVFS interfaces of modern Intel and ARM CPUs to carry out software-based attacks. These attacks take advantage of the separation of energy management from security, as DVFS interfaces can allow a user to lower the voltage or increase the frequency to a point where correct operation can no longer be guaranteed. The research we survey here targets TEEs, which can protect against other attacks (e.g., RowHammer attacks) [157]. In this subsection, we focus on the attacks that do not lead to DoS. However, it should be noted that the incorrect CPU operation resulting from these exploits may still be manifested as device unresponsiveness or lock (e.g., for a ransom) [159]. The five generic steps of DVFS-based attacks are as follows [173, 174, 206]:

(1) Environment setup: Setting up the initial voltage and frequency (which cause no faults) and clearing the residual states, which could affect the attack. These states include the cache and the branch prediction table contents, cleared by flushing the cache and running the target threads (the victim and the attacker) multiple times. The system interrupts can be disabled to avoid any external interference.
(2) Profiling for an anchor: Determining when the victim code starts executing, through software monitoring or cache side channels.
(3) Pre-fault delaying: Delaying the fault injection until the execution of specific victim instructions, to carefully tune the fault injection for exploitable hard-to-detect errors. The delay depends on the processor frequency (known to the attacker) and the executed instructions (known when targeting a standard encryption algorithm).
(4) Fault injection: Lowering the voltage or increasing the frequency to inject the desired fault.
(5) Restoring the original voltage and frequency: Restoring the normal operation of the victim.

Frequency. In 2017, Tang et al. demonstrated the first DVFS-based attack on ARM TrustZone [206]. In this so-called *CLKSCREW* attack, ⑩ in Figure 5, the adversary increases the frequency of the processor beyond the correct operation limits. The resulting faults were used to extract secret cryptographic keys from TrustZone and to escalate the privileges of a malicious kernel driver on an ARMv7 processor inside an Android mobile device. The authors created a custom kernel driver, which launches separate threads for the attacker and the victim codes, and pins each thread to a separate core. On ARM processors, different cores can have different frequencies. Hence, the attack frequency is not restricted by the effects it may cause to the adversary core. No similar attacks have been demonstrated on Intel CPUs, as many of them do not support overclocking [174].

Voltage. DVFS interfaces control the frequency and the voltage, ⑪ in Figure 5. Targeting an ARMv7 processor with TrustZone, the *VoltJockey* attacker extracts cryptographic keys and faults the RSA signature verification using a malicious kernel module [173]. Even though the voltage is the same for all cores, given that the frequency can be different, it is possible to ensure that only the victim core experiences faults. The vendor-specified operating performance points (OPPs)—specific voltage and frequency combinations to guarantee correct operation of the processor—pose no restriction for the attack, as the frequency and voltage regulator drivers can modify the frequency and voltage at the request of the top-level software. To avoid the limitation of the defined voltage threshold in the regulator property description file, the VoltJockey authors modify the voltage probe function in the voltage regulator driver. As an extension of their work on ARM TrustZone, Qiu et al. demonstrated the VoltJockey attack on Intel SGX [174]. Similarly, Murdock et al. and Kenjar et al. presented the *Plundervolt* and *V0ltpwn* exploits [99, 157]. The three works targeted some cryptographic code as part of their proof-of-concept (AES for Plundervolt and VoltJockey). The adversary requires root privileges over the system, which is not inconsistent with the use of SGX enclaves, which normally protect against processes at a higher privilege level. The main difference with respect to the TrustZone exploit is that the voltage and frequency are shared among the cores. If not carefully controlled, the low voltage may result in a kernel panic, leading to the system freezing or rebooting [174]. The voltage is controlled through an undocumented Intel model-specific register (MSR); bits of MSR `0x150` can be changed, representing the desired voltage offset, and specifying to which planes the voltage change is applied (CPU core, GPU, cache (core), uncore or analog I/O) [157]. Techniques such as *SGX-Step* [31] can be used to carefully control the fault injection against the enclave [157]. Murdock et al. investigated the voltage decline time, the possibility of faulting multiplications inside the enclave, overvolting, and differences among CPUs. Plundervolt was also used to cause the memory safety violations through faulty `imul` instructions leveraged to redirect the trusted white list into the attacker-controlled memory [157].

Kenjar et al. added an offline analysis step to test the operational limits of a processor similar to the target [99]. They also added a step to analyze the cores of the target system to determine the fault pattern and find an advantageous placement for the victim. The environment setup step included disabling the operating system drivers that communicate with the hardware interfaces for changing the voltage and disabling the automatic hardware-based selection of the voltage and clock frequency. Finally, Kenjar et al. introduced the concept of *stressors*—pieces of code running on the logical partner of the victim's core to increase the resource contention and the core temperature for an improved likelihood of fault injection. Hadjilambrou et al. [82] and Kim and John [103] show that highly effective stressors can be automatically generated using genetic algorithms for the purpose of an attack or voltage noise characterization on CPUs.

4.3.3 *Countermeasures.*

Processors typically handle load transients by detecting core voltage droops and then decreasing the working frequency [57]. *Razor* latches inserted in the pipeline are

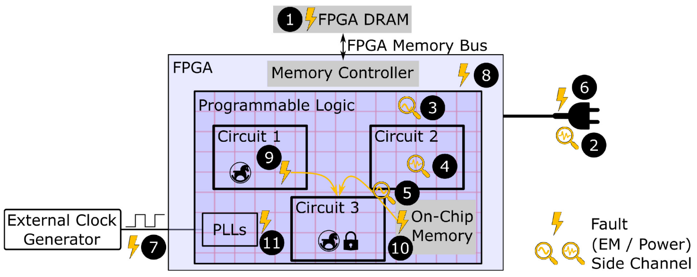

Fig. 6. An illustrative FPGA-based system, with highlighted electrical-level vulnerabilities.

also helpful; they sample the output with two phase-shifted clocks to detect discrepancies and repeat computations to fix them [55].

Checking for faults and attempting to recover from them works regardless of how the fault was injected (software or hardware interfaces). Aumüller et al. developed a software countermeasure for the RSA signature on a smart card with no hardware countermeasures [9]. They relied on the fact that voltage spike-induced faults can be injected into any value except those from a read-only memory, which can be used to check computation correctness. Redundancy is one of the most common fault detection and recovery defenses. Moro et al. evaluated two software-based countermeasures against EM-fault injection, both aiming to detect or tolerate single faults, or specifically those which can cause instruction skipping [152]. One countermeasure repeats the instructions, while the other executes an instruction twice, storing the results in different registers and then comparing them. Both defense strategies improve the security for their specific use cases (a subset of instructions to protect or only protecting against instruction skipping) but do not render the executed code fully secure. Making the exploitation of the fault-injection interfaces (e.g., DVFS) more difficult for the adversary is also a possibility. In response to the software-based fault injection against SGX, Intel has disabled the overclocking mailbox interface configuration [157]. The perfect countermeasure does not exist, but system designers and programmers can choose to deploy some of the available countermeasures for security-sensitive code.

5 ELECTRICAL-LEVEL ATTACKS ON FPGAS

The FPGA, being both a prototyping platform and an acceleration platform, has received considerable attention from the hardware security community. Similar to CPUs, electrical-level attacks are applicable to FPGAs. In particular, the low-level programmability of FPGAs allows adversaries to exploit electrical phenomena for their attacks [144]. We denote here the attacks carried out via the bitstream (i.e., FPGA configuration file) as software-based; the bitstream file is used to program the basic hardware blocks of the FPGA and have them obtain desired functionality. In addition to software-based attacks, we survey and discuss the attacks that target soft processors, common in embedded FPGAs. We show an example of an FPGA-based system, with highlighted vulnerable interfaces, in Figure 6.

5.1 RowHammer

While the experimental setup by Kim et al. for studying RowHammer bitflips was based on FPGAs [102], for a long time, it was CPUs that RowHammer attacks were targeting. Recently, however, Weissman et al. examined whether FPGAs could be used for accelerating RowHammer exploits [224]. To that purpose, they used an Intel Arria 10 *programmable acceleration card* [8]. Their FPGA-based RowHammer attack, termed *JackHammer*, turned out to be significantly faster than the RowHammer exploit launched from the CPU, because FPGAs do not need to flush the memory addresses. The attack targeted a WolfSSL RSA application, running on the host CPU. While in JackHammer the FPGA acts only as the enabler of the exploit, the fact that it *can* induce faults in the shared memory means that FPGAs can also be the victims of JackHammer attacks. We underscore this exploit as ① in Figure 6.

5.1.1 Countermeasures. The most effective countermeasures against RowHammer, which would protect even against an FPGA-based attacker, are those implemented in the DRAM or the memory controller. Other countermeasures that rely on the software running on the CPU are not suitable unless they are adapted to work on the FPGA's hardware [54]. In the absence of hardware performance monitors on the Arria 10 platform, Weissman et al. proposed implementing them on the FPGA [224]. Elnaggar et al. used a monitor on the FPGA fabric to detect attempts to launch RowHammer exploits [54]. Weissman et al. also proposed defense mechanisms specific to the tested platform. The current version of Open Programmable Acceleration Engine (OPAE), used on the Arria 10 platform, makes physical addresses available to the user space, and allows for huge pages. Disabling huge pages and virtualizing the addresses would render finding eviction sets considerably harder, thus improving system security. Algorithmic-level patches may also be useful to protect against the fault injection attacks against RSA [224]. However, without these countermeasures actually being deployed and evaluated, the security of current systems against RowHammer remains questionable.

5.2 Side-Channel Analysis

Many side-channel exploits have been demonstrated on FPGAs, most often relying on the attacker having hardware access to the victim device. However, hardware-security researchers have recently taken advantage of the programmability of FPGAs to carry out remotely controlled side-channel attacks, targeting the FPGAs hosted by the cloud service providers. In the following subsections, we discuss both types of exploits.

5.2.1 Hardware Side-Channel Attacks. Similar to the exploits against CPUs, an attacker with access to the hardware can measure the power consumption or the EM emanations from an operational FPGA and gain information about the algorithm executing on the device.

Power. Power side-channel attacks against FPGAs, ② in Figure 6, were first demonstrated in 2003 [162]. Soon after, Standaert et al. examined how DPA differs when applied to FPGAs versus CPUs [200]. Since then, many works have successfully demonstrated hardware SCA on FPGA implementations of cryptographic algorithms [20], sometimes helped by hardware Trojans [114, 117]. Researchers also looked at how to optimize side-channel exploits, for example, by finding the optimal linear transform to apply to the traces to more easily distinguish the correct key [163]. Yao et al. demonstrated SCA on a masked AES executing on a soft RISC-V processor, implemented on a Xilinx Spartan-6 FPGA [226]. At first, they collected the side-channel traces to distinguish between various algorithmic steps (e.g., mask generation or encryption). Then, with a carefully controlled clock glitch fault injection (⑦ in Figure 6), they managed to disable the mask and recover the secret key using standard first-order SCA. FPGA cryptographic circuits are not the only targets of the power SCA attacks. The encrypted bitstreams, which should protect the confidentiality of proprietary designs, are vulnerable as well. Moradi et al. successfully broke the bitstream encryption of an Intel Stratix-II and a Xilinx Virtex-II FPGA through power SCA, where power measurements are facilitated by inserted resistors on the paths of interest [148, 149]. While power-analysis exploits focus on the dynamic power consumption, Moradi used static power consumption against AES FPGA implementations with masking and shuffling countermeasures [147]. Even though the attack is feasible, the signal-to-noise ratio (SNR) of the static power is considerably smaller than that of the dynamic power, thus requiring a particular measurement setup. Furthermore, the adversary model is much stronger, assuming control over the clock signal. Finally, while not targets for disassembly, FPGAs can be programmed to implement softcore processors. Power traces collected in that case allow for side-channel disassembly [113].

Today, FPGAs are commonly used for accelerating ML algorithms, which are an attractive target for attackers. Wei et al. carried out a power SCA on a CNN implemented on the SAKURA-G FPGA board [222]. After collecting power traces, they applied a noise-removal technique and then carried out two input-recovery exploits. The first assumed a passive adversary who tries to distinguish pixels that are part of the input image background. The second assumed an adversary capable of sending inputs to the target for profiling purposes who, using prebuilt power templates, tries to recover the pixels of the input image [222]. Dubey et al. used DPA to extract the binary neural network (BNN) model parameters (weights and biases) [49]. In addition to being the targets, ML algorithms are enablers of the exploits: Mukhtar et al. used ML to recover the secret key of an elliptic curve cryptography algorithm running on a Xilinx Kintex-7 FPGA [154], while Wang and Dubrova combined several CNN classifiers to reduce the number of traces necessary to recover the AES encryption key on Artix-7 FPGAs [220]. Ramezanpour et al. used unsupervised learning against AES [178], while Aydin et al. leveraged deep learning for breaking the lattice-based key-exchange protocols [11].

EM Field. EM emanations are also useful for side-channel analysis on FPGAs [34]. Carlier et al. and De Mulder et al. used the EM SCA, ③ in Figure 6, to attack an AES FPGA implementation [34] and break the security of an FPGA implementation of an elliptic curve cryptosystem [155], respectively. Similarly, Kim et al. carried out the power and EM SCA on the ARIA block cipher running on an Intel APEX 20K FPGA [100]. Investigating the strengths and limitations of the EM SCA, Heyszl et al. showed that the SNR changes with the location of the EM probe. Due to local leakage, better attack results can be obtained if the probe is positioned over the target part of the design after a profiling phase [86]. Hori et al. carried out correlation-based SCA against an AES running on Xilinx Kintex-7 and Virtex-5 FPGAs [88]. With the smaller technology node of the Kintex-7 family, fewer traces were required for the full recovery of the cryptographic key. The SNR of subkeys at some locations was different than at others, indicating that the structure of the circuit does affect the attack success [88]. Bitstream encryption is vulnerable to EM side-channel analysis: Moradi and Schneider demonstrated the attack feasibility on Xilinx series 4, 5, 6, and 7 FPGAs [150]. ML algorithms also fall victim to EM SCA: Yoshida et al. used correlation EM analysis to recover the weights of an MLP under the assumption that the adversary has control over the inputs to the MLP [230].

5.2.2 *Software-Based Side-Channel Attacks.*

By carefully programming the FPGA logic, a malicious party can implement an on-chip sensor for measuring the core voltage variations or the crosstalk-coupling between neighboring long wires. These software-defined sensors are the enablers of remotely controlled attacks, targeting cloud FPGAs.

Power. Remote power-side channel attacks, ④ in Figure 6, commonly employ sensors based on ROs, easily created by connecting an odd number of inverters in a loop. The RO frequency of oscillation depends on the loop delay, which, in turn, depends on the power supply voltage. To monitor its variation with time, it is sufficient to connect the RO output to the clock input of a frequency counter and to sample the counter output at fixed periods of time.

Zick and Hayes were among the first to use ROs for monitoring on-chip variations [238]. To minimize the sensor footprint, they employed residue number system ring counters using shift register look-up tables (LUTs), resulting in only 8 LUTs per sensor on a Xilinx Virtex-5 FPGA. On a Zynq-7020 SoC, Zhao and Suh showed that ROs are the enablers of side-channel attacks against RSA cores and CPUs on the same SoC [236]. Their insight was that a power side channel allows for distinguishing the CPU activity levels, suitable for timing-based attacks against RSA exponentiation operation, which executes in constant time. One drawback of RO sensors is that, to obtain good measurement resolution, the sampling period of their frequency counter should be hundreds or even thousands of clock cycles. To address this limitation, Gravellier et al. combined high-frequency ROs with Johnson ring counters [75].

As an alternative to limited-resolution RO sensors, Zick et al. proposed delay line-based time-to-digital converters (TDCs), capable of sensing nanosecond-scale voltage transients on FPGAs [239]. Many subsequent works employed TDCs: Schellenberg et al. used TDCs for a CPA attack against an AES FPGA core [189] and for an inter-chip power side-channel attack [190], showing that remote attacks are not limited to a single chip only.

Remote side-channel exploits have been demonstrated in the cloud setting as well. Glamočanin et al. were the first to demonstrate a successful power SCA attack against an AES core, using TDCs, on Amazon AWS F1 instances [71]. More recently, Moini et al. used TDCs to recover the inputs to a BNN, under the assumption that the adversary knows the BNN structure [146]. Zhang et al. used TDCs to remotely steal the structure of an FPGA neural network [234].

EM Field. The possibility of remote measurement of the EM field emanated from an FPGA core is an open research question. However, the effects of the EM coupling between neighboring FPGA wires (⑤ in Figure 6) can be measured remotely. It has been shown that the EM coupling between neighboring FPGA wires can leak information about the logical level carried by the observed wire [60, 69]. Ramesh et al. used an RO sensor to learn the AES final round key. First, they ensured that the RO is routed next to one of the long wires carrying inputs to the final round Sbox. Then, with the observed ciphertext (assumed to be publicly available) and the oscillation count of the RO, they examined all possible values for the target key byte until finding one consistent with the obtained measurements [177]. Giechaskiel et al. showed that a variety of RO-based crosstalk-coupling sensors can be successfully deployed on cloud FPGAs while bypassing the checks for combinational loops put in place by the cloud service providers [67].

5.2.3 Countermeasures. Protections against SCA on FPGAs focus on decreasing information leakage or preventing remotely controlled attacks by disallowing sensor circuits or better isolating the neighboring designs. Two common techniques for decreasing information leakage are hiding and masking [133]. In hiding, the goal is to reduce the SNR available to the adversary. In masking, the data processed by the circuit is obfuscated so that the observable leakage is independent of the secret [181]. Implementing these techniques incurs considerable area or frequency penalties. Furthermore, the circuit can still remain vulnerable to more advanced, higher-order attacks [133].

Le Masle et al. used a network of on-chip RO sensors and a proportional-integral-derivative controller to monitor and maintain an approximately constant core voltage, effectively reducing the SNR [135]. As actuators, they employed long routing wires (equivalent to high capacitive load). In a similar direction, Krautter et al. used a TDC sensor to control a noise-generating *fence* composed solely of ROs [110].

Güneysu and Moradi proposed and evaluated a set of countermeasures against DPA on FPGAs [81]. They generated Gaussian noise by leveraging shift register LUTs, BRAM write collisions, and short circuits in the switch boxes. They randomized the clock signal to impede data alignment and implemented a circuit to detect whether the external clock is manipulated. Finally, they proposed scrambling the contents of Sboxes or Tboxes. These countermeasures were tested on an AES core in a Xilinx Virtex-II Pro FPGA.

Sasdrich et al. proposed leveraging the dynamic reconfiguration capability of FPGAs to dynamically change the hardware implementation of a PRESENT cipher and thus improve the resistance against SCA exploits [187]. Dubey et al. investigated the application of masking and hiding to a BNN, demonstrating improved security but also highlighting the challenges of building a fully side-channel resistant neural network [49].

To detect FPGA sensors, Krautter et al. [112] and La et al. [115] developed bitstream scanners, which reconstruct the design netlist and search for so-called *signatures* of potentially malicious circuits. However, the developed scanners target only a subset of FPGA families: Lattice iCE40 [112] and Xilinx Ultrascale+ [115].

Mitigating crosstalk side channels was first addressed by Huffmire et al., who proposed using *moats* (guard bands between design partitions) for spatial separation and *drawbridges* for routing core-to-core communication signals and isolating them [92]. Yazdanshenas and Betz proposed wrapping user applications into soft shells and encrypting incoming and outgoing data [228]. Compared to unencrypted traffic, they report up to 80% higher latency and a 20% decrease in the available user area. Luo and Xu developed HILL, a hardware isolation framework, which places security-critical nets in the center of a core, routes them without using the long wires, and isolates them from other FPGA cores [127]. Seifoori et al. improved the FPGA compilation to ensure that the post place and route designs are protected by construction, that is, that no untrusted signal is routed close to a security-critical net [194].

5.3 Fault Injection

An adversary with hardware access to an FPGA can externally inject faults. Moreover, the programmability of FPGAs allows for injecting faults internally. We discuss both scenarios here.

5.3.1 Hardware Fault Injection.

Voltage. FPGA circuits are sensitive to both undervolting and overvolting [33] (⑥ in Figure 6). Undervolting increases circuit delays, leading to timing violations, whereas overvolting glitches cause transient oscillations in the internal voltage while trying to recover a nominal voltage value, ultimately resulting in transient undervolting [241]. Bhasin et al. evaluated the effects of undervolting on several AES implementations and observed that, when no countermeasures are implemented, the hardware with the lowest area is also the most vulnerable to differential fault analysis [21]. Zussa et al. investigated the mechanism of injecting faults into FPGA circuits through voltage glitches and concluded that it is the same as with clock glitches [241].

Frequency. Violated timing constraints can also be a consequence of clock glitches, ⑦ in Figure 6 [2, 138, 231]. Agoyan et al. carried out a DFA attack against an AES circuit running on a Xilinx Spartan 3AN FPGA. With another FPGA serving as the external clock generator, they used a delay-locked-loop to generate two clock signals (one of them delayed). A glitch was injected when combining the rising edge of the delayed clock with the falling edge of the regular clock. This reduces the setup time available before the next clock edge [2]. Korczyc and Krasniewski used two clocks, a high-frequency and a low-frequency one, to generate glitches and evaluate the susceptibility of FPGA circuits to clock-glitching attacks [109]. Glitches were created by alternating between these two clocks, as the resulting clock would have some periods shorter than others.

Yuce et al. targeted a LEON3 RISC processor [231]. First, they showed that clock glitches can lead to faults that are not mitigated by instruction duplication, triplication, instruction parity, and a countermeasure against skipping instructions. Second, they leveraged this knowledge to break a fault-resistant implementation of an LED block cipher.

EM Field. FPGA hardware is also vulnerable to EM fault injection, ⑧ in Figure 6. Dehbaoui et al. used EM fault injection to attack unprotected and protected FPGA AES cores. They showed localized effects of the fault injection and, hence, the countermeasure, which was a timing violation detection circuit, was shown to not be consistently effective [46]. Madau et al. used EM-induced faults to attack a true random number generator as a first step to overcoming the masking countermeasures for cryptographic cores [129].

5.3.2 Software Fault Injection. Recent work has shown that valid FPGA bitstreams can be used to create core voltage fluctuations [73] with the goal of resetting the FPGA (a DoS attack) or causing a computational fault (a fault attack). Additionally, the reconfigurable clock generation circuits can be programmed to inject faults.

Voltage. ROs are not only able to sense on-chip delay changes, they can also be used as power viruses (the Trojans denoted with ⑨ in Figure 6). When instantiated in large numbers and enabled periodically, they can cause voltage drops capable of sending an FPGA to reset [73] or injecting a computational fault. Krautter et al. used ROs for a differential fault analysis attack on an AES core [111], while Mahmoud et al. used them to bias self-timed true random number generators [130] and to remotely trigger a stealthy hardware Trojan (Circuit 3 in Figure 6) [131].

Given that combinational ROs are the enablers of a multitude of remote electrical-level attacks on FPGAs, some CSPs (e.g., Amazon AWS) prevent them from being synthesized. However, ROs are by far not the only circuits capable of high power consumption; Alam et al. used dual port FPGA RAM memories (⑩ in Figure 6) to create short circuits, causing severe voltage drops. They leveraged the induced faults to attack a finite state machine and to cause a neural network to misclassify [5]. Provelengios et al. evaluated several power-wasting circuits, all allowed by the CSPs [170]. These included two designs in which FFs are inserted into the RO loop. In the first, the FF was clocked by a PLL-generated clock, while its data input comes from the RO inverter. The second design uses the RO inverter output to clock the FF and the input of the inverter to clear the FF. Finally, they demonstrated that shift registers and AES rounds (which violate timing constraints) can serve as power viruses. Matas et al. used multiple-input XOR gates to generate glitches and then amplify the glitches by having them charge long routing paths [137]. Boutros et al. tested other power-wasting circuits: asynchronous ROs, clock-gated garbled XORs, and clock-gated hybrid logic [26]. The garbled XORs are a collection of multiple-input XOR gates, with inputs connected to the outputs of toggling registers. The hybrid logic combines the garbled XORs with block random access memories (BRAMs) and digital signal processing (DSP) blocks. In their work, these attacker circuits were clock-gated to cause large current changes the moment they are enabled.

Frequency. Many works that investigate the effect of clock glitches on FPGA circuits take advantage of the programmable fabric to build the glitch generator. Hence, the exploit itself does not require hardware access to the FPGA. These exploits (⑪ in Figure 6) are a threat if a hardware Trojan is present in the design, as each FPGA application can have its own clock generated from a corresponding PLL otherwise. Bonny and Nasir used PLLs to create clock glitches and attack an FPGA implementation of a chaotic oscillator [24]. Ghalaty et al. leveraged the PLL on an Intel Cyclone IV to generate clock glitches for a differential fault intensity analysis in which the attacker varies the intensity of the fault injection and observes how the faulty behavior of the circuit changes [66]. Since the change is secret dependent, it can be leveraged to recover the secret. Additionally, the clock glitches have been successfully used to recover the secret key of an AES module [121] or cause a neural network to misclassify [122].

5.3.3 Countermeasures. While relaxing the timing margins is one way of decreasing the risk of fault injection exploits, it does not eliminate the risk altogether. Therefore, other countermeasures are needed, for example, for detecting and correcting the faults or for detecting the presence of fault-injection circuits (to stop them or notify the victim).

The main contributor to circuit delay variations on FPGAs are voltage droops caused by fast load transients [73, 143]. Shen et al. designed a voltage droop detector for an Intel Cyclone IV FPGA, which uses an adder to measure the propagation delay of a carry input. If the propagation delay is above a specified threshold (calibrated in advance), a voltage droop is signalled. This signal is used to suppress the next clock edge, thus avoiding the injection of a timing fault [197]. Focusing on cloud FPGAs, Provelengios et al. [169] and Mirzargar et al. [143] proposed distributed sensing of on-chip voltage variations for detecting and locating the origins of voltage droops. Provelengios et al. used a regular network of RO-based sensors on a DE1-SoC board to reconstruct voltage contours and locate the source of the high power consumption [169]. Mirzargar et al. developed a fully automated approach for Xilinx FPGAs, in which the RO sensors are implemented after design placement and using free resources only [143].

Following the approach of Razor [55], Stott et al. built a timing-fault detector using a shadow register, a comparator, and a latch to signal whether a fault has occurred. These fault detectors can also be automatically inserted into arbitrary FPGA applications, as needed by the timing report [202]. Many other works built similar sensors to detect that a fault has occurred or to protect a target implementation [26, 131]. Zussa et al. built a clock glitch detector to defend against EM fault injection. Given the local effects of EM fault injection, they needed to deploy a network of sensors on-chip [240]. Jiang et al. proposed a solution for measuring the delay of a desired path, which may not necessarily end in a register against which to compare [95]. Their design, useful for measuring delays of paths ending in BRAMs or DSP blocks, has two registers, clocked by the last signal in the path (which would act as the register's input in other designs). The two registers sample a shifted clock as their input. The outputs of the two registers are passed through an OR gate, whose output is monitored to return a numerical value for the path delay. Focusing on clock glitches, Luo et al. generated a high-frequency signal using ROs and used it to monitor the clock and detect when glitches are present [126]. This is only one of many examples of ROs being used for protecting instead of attacking.

Aghaie et al. provided a full fault coverage when an adversary is assumed to be able to inject faults into a number of signals during each clock cycle [1]. Their methodology combined the use of checkpoints with duplication. While each countermeasure has its own overhead, the combination proves to be efficient in detecting faults, with a reasonable overhead. Luo and Xu proposed a frequency scaling-based approach, in which they build a framework to dynamically change the application frequency (using feedback information from an on-chip delay-line sensor) and avoid the occurrence of timing faults [128]. For EM fault injection, Miura et al. successfully employed on-chip PLLs, which act as monitors of the clock's stability, for attack detection [145]. An RO was routed around the core to protect, and its output acted as the monitored signal. Once that signal experiences a sudden change of frequency due to EM fault injection, the PLLs stop being locked and the attack is detected.

Finally, fault-attack countermeasures can focus on optimizing the architecture of future devices. Ahmed et al. designed an optimized LUT with input-to-output delays less sensitive to supply voltage changes. Motivated by dynamic voltage scaling, they decoded the slowest two inputs of the LUT and used separate voltage islands for LUTs and routing [3].

6 ELECTRICAL-LEVEL ATTACKS ON GPUS

GPUs have also recently received attention with regards to their security. Being similar to CPUs, they suffer from the same vulnerabilities. That being said, the literature on electrical-level attacks

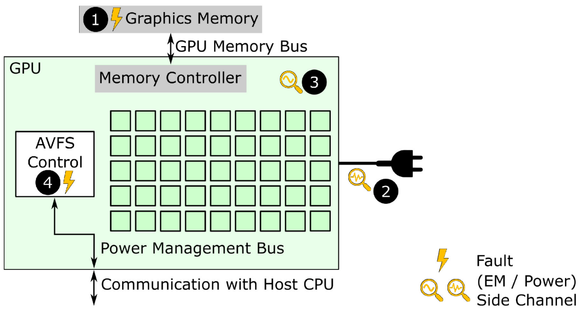

Fig. 7. Representative GPU system, with highlighted vulnerabilities.

on GPUs is sparser and more recent than for CPUs. In this section, we discuss the demonstrated exploits that target or leverage GPUs.

6.1 RowHammer

Frigo et al. leveraged GPUs to accelerate RowHammer exploits [58] and allow an adversary, controlling a malicious website or advertisement on the website, to escape a browser sandbox and gain arbitrary read/write privileges on an Android platform. To find the target main memory physical addresses, they built efficient eviction sets, achieved contiguous memory allocations, and exploited a timing side channel to obtain their locations. This type of attack, ① in Figure 7, is capable of affecting the entire heterogeneous system.

6.1.1 Countermeasures. The protections against RowHammer exploits are most efficient if implemented in the main memory or the memory controller. The exploit by Frigo et al. could be thwarted by better memory partitioning to ensure that the victim memory rows are not reused to store sensitive data [58]. However, the trade-off of this countermeasure's complexity and security is not yet evaluated [58]. Even if this countermeasure does not fully eliminate the RowHammer vulnerability, it makes the exploits more challenging.

6.2 Side-Channel Analysis

In a similar fashion to CPUs and FPGAs, GPUs can leak information through side channels, even though the massively parallel code execution results in somewhat different requirements for the side-channel analysis. The works we discuss here show that hardware-based SCA attacks on GPUs are a real threat.

6.2.1 Power. Luo et al. implemented a power SCA attack (② in Figure 7) against an AES encryption running on an NVIDIA TESLA GPU to extract the last round key using CPA [125]. They present their methodology for a successful attack, even under the concurrent execution of a vast number of processes on GPUs, and the noise in the collected traces. Cronin et al. chose a different target—a mobile platform with CPUs and GPUs, responsible for processing graphics on smartphones [43]. They found that information leaks through the USB charging interface and used it to break a passcode typed while the smartphone was charging.

6.2.2 EM Field. An adversary with physical access can measure the EM field created by the GPU, ③ in Figure 7. With an EM-based SCA attack, Gao et al. successfully broke the security of a bitsliced AES implementation on GPUs otherwise immune to cache-based side channels [62]. Gao and Zhou targeted single-instruction multiple threads (SIMT) execution on CUDA-enabled GPUs, accelerating an SBox LUT-based AES [61]. In their experimental setup, all threads were executing different encryptions with the same secret key, but the adversary could choose some of the threads to encrypt the same plaintexts. Under those conditions, they demonstrated a successful key recovery exploit.

6.2.3 Countermeasures. The concept of masking is not only applicable to CPUs and FPGAs, but also to cryptographic primitives executing on a GPU. However, masking can quickly become too complex due to the required generation of random masks [61]. A simpler approach would rely on restricting attacker capabilities, knowing that side-channel exploits against GPUs rely on chosen-thread attacks, in which the attacker can choose multiple threads to encrypt the same random plaintext and seeing that no exploits with random plaintexts have been demonstrated yet [61]. To characterize GPU power consumption, Mukherjee et al. built *GIPSim*, a model of power side-channel leakage [153]. Such models are a foundation for future protection frameworks, which could select the instructions to execute considering the primitive to protect. The concurrent execution of the primitive and the instruction would reduce the SNR, rendering the exploit significantly more difficult.

6.3 Fault Injection

6.3.1 Software-Based Fault Injection. For lowering power consumption and maintaining high performance, modern GPUs provide AVFS interfaces. Exploits utilizing AVFS are highlighted as ④ in Figure 7. Sabbagh et al. leveraged AVFS on the AMD Polaris GPU to inject random faults into its execution. The authors exploited Radeon Open Compute—system management interface (rocm-smi) tools and application programming interfaces to set the OPPs for their exploit and the timing to activate them. The adversary can send the fault injection commands from the host CPU to the GPU, resulting in exploitable faults for DFA on AES [185]. Gongye et al. launched the same exploit against a ResNet-18 model to reduce its classification accuracy [74].

6.3.2 Countermeasures. Sabbagh et al. discuss potential defenses against their demonstrated attack. These include limiting the OPPs to ensure that OPPs that lead to faults are not allowed. Recognizing operating limits requires characterizing the GPU hardware, which may need to be done periodically to account for aging effects. Increasing the guardbands can also reduce the possibility of fault injection [185].

7 IMPLICATIONS FOR HETEROGENEOUS SYSTEMS

When CPUs, FPGAs, and GPUs are brought together in a heterogeneous computing system, new electrical-level exploits may arise. In this section, we provide our insights on whether the attacks known to affect one computing component may pose a threat (i.e., extend) to other components in a heterogeneous computing system and how the defenses should be adapted. To that goal, we proceed as follows:

- Starting from the surveyed attacks on CPUs, FPGAs, and GPUs, we identify those that can affect more than one component given the heterogeneous system architecture models presented in Section 2.
- If applicable, we examine how the threat model changes in a heterogeneous use case.
- We discuss the proposed and deployed countermeasures to assess whether they remain relevant for the HCSs.

Through this evaluation, we highlight the open research directions for the security of HCSs against electrical-level attacks. These future research directions, summarized in Table 1, range from investigating and understanding novel vulnerabilities to designing appropriate countermeasures and deploying software- and hardware-defense strategies. Designing and deploying secure HCSs is

Table 1. Open Research Directions

Attack	Access	Target	Open Research
RowHammer	SW	FPGA	Vulnerability to attack and adapted countermeasures
		GPU	Vulnerability to attack and adapted countermeasures
		HCS	Deploying secure HCSs
SCA	HW+SW	FPGA	Countermeasures against SCA sensors
	HW	CPU	Countermeasures against disassembly
	SW	CPU + HCS	Vulnerability to remote disassembly and countermeasures against it
	SW	HCS	Intercomponent leakage and countermeasures
	SW	HCS	SCA through monitors and countermeasures
Fault Injection	HW+SW	FPGA	Countermeasures against power wasters
	SW	HCS	Intercomponent fault injection and countermeasures
	SW	HCS	Combining DVFS/AVFS with power wasters and countermeasures
	HW+SW	HCS	Building new fault models
	HW+SW	HCS	Deploying HCSs secure against fault injection

HW: Hardware access; SW: Software access; HCS: Heterogeneous Computing System.

not limited to one use case. Therefore, the solutions should consider various deployment models, including embedded devices and virtualized accelerators, as well as various attacker models (e.g., a malicious cotenant or a malicious CSP). While CPUs and GPUs can be virtualized, FPGAs have not been virtualized yet (even though there is considerable research on FPGA virtualization [176]). While virtualization can hinder some exploits by abstracting away details of the hardware and restricting access to some hardware-dependent interfaces, it does not necessarily provide full protection. For instance, if the adversary has hardware access, attacks that bypass the virtualization remain feasible [30]. The situation is not very different when the attacker has software access (e.g., RowHammer remains feasible [180]). Consequently, virtualization of heterogeneous computing systems, in view of electrical-level vulnerabilities, remains an open research direction.

7.1 RowHammer

As discussed in Section 2, in an embedded platform, the main memory is typically shared between the components. Furthermore, in heterogeneous systems, GPUs and FPGAs can typically have access to the main memory (DRAM) of the host CPU. Researchers have shown that this access can be used to accelerate RowHammer attacks [58, 224]. Moreover, the FPGA's and GPU's access to the DRAM also means that faults in the memory may affect the computation of any of the components that use the faulty value. We underscore this threat in the first two entries in Table 1.

RowHammer attacks typically target the CPU to gain unauthorized privileges or to fault the operation of security primitives; as such, most RowHammer works focused on the CPU as the victim. Depending on the code running on the GPU or the hardware deployed on the FPGA, faulting security primitives may be the goal of the RowHammer attack against a GPU/FPGA, or new exploits can arise according to the use scenario, which remain to be demonstrated. For instance, if an encryption circuit (e.g., RSA) is implemented on the FPGA, its stored key values may be faulted, leading to incorrect computations by the FPGA circuit. Similarly, fault attacks can be carried out against ML models to cause accuracy degradation similar to what was demonstrated by Hong et al. [87]. Therefore, RowHammer attacks likely do pose a threat to an HCS as a whole. Furthermore, since the DMA is accessible not only to the internal components but also to the I/O interfaces, RowHammer attacks requiring no process residing on the system (e.g., Nethammer [120] and Throwhammer [207]) are another threat, to be investigated for FPGAs, GPUs, and HCSs.

In a heterogeneous computing system, the adversary can reside on and try to attack any of the system components, and virtualization does not necessarily provide protection [180]. Furthermore, the attacker goals may be more varied according to the computation that is targeted. This changes the threat model significantly, as it no longer holds that only the CPU interfaces need monitoring. The threat model change also requires reexamining the countermeasures. As an example, let us consider one of the main enabling features of the accelerated RowHammer attack by Weissman et al.: the lack of hardware performance counters (to monitor the FPGA's accesses to the memory) [224]. One could argue that, because multitenancy is not yet a reality, there should be no security concerns and, hence, no need to deploy the countermeasures in that case. However, the potential existence of Trojans and the possibility of multitenancy in the future render the search for and the deployment of effective countermeasures important (third entry in Table 1).

While all proposed countermeasures have their disadvantages (power and performance overheads, memory overhead, lack of backward compatibility), they can still render the exploit significantly harder. Furthermore, though most proposed countermeasures are concerned with only CPUs, their principles can be extended to the memory accesses of GPUs and FPGAs. Many of the proposed countermeasures focus on existing CPU systems and, hence, require no modifications to the hardware (e.g., ZebRAM [106], which uses guard rows to protect against RowHammer bitflips). In this case, research can show how to adapt them to other system components using existing drivers and software interfaces. Some of the countermeasures, however, require the existence of hardware performance counters (e.g., ANVIL [10], which uses the counters to identify frequently accessed rows). Taking advantage of the flexibility of heterogeneous platforms, performance counters may be implemented in the programmable logic (similar to the idea of Elnaggar et al. [54]).

Depending on the data or the computation to be protected, countermeasures may even focus on the algorithm or on protecting only part of the memory (e.g., GuardION [216], which protects against DMA-based exploits on mobile devices by isolating DMA buffers with guard rows). For protection against DMA-based exploits, transaction control, similar to what is available on the Xilinx UltraScale+ MPSoC [242], may prove useful for limiting the access rate of potentially malicious channels. However, further investigation is required to assess the effectiveness and the required modifications for the proposed countermeasures when used in heterogeneous platforms. To effectively test the countermeasures on heterogeneous platforms, the threat model for a heterogeneous system (considering all of the possibilities of how an adversary can gain access to the system) should be considered. This should be followed by an evaluation of the effectiveness of a RowHammer exploit. Based on this investigation, stronger recommendations can be provided on how to design heterogeneous computing platforms that are secure against RowHammer exploits.

7.2 Side-Channel Analysis

Side-channel exploits that rely on the physical access to the device are a real threat for all three examined system components. Therefore, countermeasures against them should be investigated. Furthermore, there is still room for more research on side-channel disassembly, for example, for processors with higher clock frequencies, in noisier environments, or for an HCS with various components executing concurrently. However, the more interesting threats—which we deem feasible and requiring further investigation—are those that can use software interfaces to carry out SCA remotely and across components.

The existence of monitors (such as the RAPL interface for Intel CPUs and the system monitor in the UltraScale+ MPSoC), which can report the consumed power of various components, is one obvious threat. PLATYPUS and similar attacks demonstrated the risk of unprivileged access to such interfaces in a CPU-based system [118, 134]. Accordingly, we extrapolate the possible risk against other HCS components. Consequently, the typical threat model of an adversary with physical access is no longer necessary for SCA. Further investigation is required to quantify the information that can be gained from power-reporting interfaces on existing embedded and cloud systems, which is why we consider secure monitoring as an open research direction in terms of describing the vulnerability in a heterogeneous setting and in terms of defenses that do not remove the benefits of the monitors. The possibility of maliciously using the monitors, which is to be investigated, changes the usual threat models for GPUs and FPGAs, which are now at risk from the CPU processes.

Another risk comes from the possibility of implementing voltage sensors in the PL of the FPGA [73]. Having been used for remotely controlled SCA against a CPU on the same chip, the open research question is whether they would enable remote side-channel disassembly. Given the tight integration in (MP)SoC platforms, in terms of sharing the power resources, the FPGA sensors are likely a higher threat for chip-level than for system-level heterogeneous platforms. Yet, as discussed in Section 2, if the voltage regulator is shared by the CPU and the FPGA (given their requirement for the same voltage level), an equivalent of the tightest level of integration is inadvertently provided. The amount of information that can be gained at various levels of system integration (e.g., same power chip, but different voltage regulators) is a topic that deserves further investigation, in particular in light of the covert communication channel demonstrated by Giechaskiel et al. [68]. Their C3APSULe exploits disclosed the possibility of covert communication between an FPGA and a CPU or a GPU when sharing the same rack-level power supply [68].

Those SCA countermeasures, which do not require hardware modifications, would likely not fully prevent information leakage. This does not mean that they are ineffective, as hiding and masking do make breaking cryptographic keys considerably more difficult and time-consuming [19, 81]. The issue is that hiding and masking focus on making the side-channel information independent of the secret data to be protected, meaning that they would not, in general, prove effective against side-channel disassembly. They might render some information harder to obtain (e.g., instruction operands), but there is no reason to assume that they would completely prevent the leakage. Hence, countermeasures against side-channel disassembly remain an open area of research, as highlighted in Table 1.

In the case of hardware side-channel disassembly, depending on the threat model, countermeasures may include detecting power probes or improving packaging to reduce EM emanations. For software side-channel exploits, countermeasures would need to limit the information leaked across components, and the information provided from the system interfaces (SCA through monitors in Table 1) to render attacks impractical, without compromising the usefulness of these interfaces. Countermeasures against data leakage should work against both hardware- and software-based secret recovery exploits, and they should be tested to understand their impact on instructions leakage. Accordingly, relying on virtualization to prevent access to system monitors is not sufficient to guarantee system security. Finally, defense mechanisms that detect the presence of sensors on FPGAs are needed to protect against an adversary residing on the programmable fabric [112]. Given that typical FPGA sensor architectures may not be the only circuits enabling side-channel information recovery [72], effective defenses are yet to be found.

7.3 Fault Injection

Similar to SCA, fault-injection attacks that manipulate the clock or the power supply to the chip are a threat to all electronic components. Further studies are required to analyze whether manipulating the power supply can lead to faults in more than a single component of a heterogeneous computing system (i.e., novel fault models should be considered; see Table 1). For example, an attack such as VoltPillager [37], which sends messages on the Intel SVID bus (power management bus in Figures 2 and 5) to change the voltage of the CPU voltage regulator, may affect one component at a time (depending on the available interfaces and the control messages sent on them). On the other hand, if all of the components share the same voltage source and that source is manipulated, then faults may occur in several of them depending on their undervolting, frequency, and timing constraints.

For remote fault injection, the existence of DVFS (and AVFS) interfaces remains a threat for CPUs (and GPUs). Voltage and frequency scaling for FPGAs—which has been considered but not yet implemented [4, 128]—would need to be carefully deployed for new vulnerabilities to be avoided. Another important threat comes from the possibility of implementing power-wasting circuits on FPGAs—their impact can extend to multiple system components, potentially enabling intercomponent fault injection, as underscored in Table 1. The demonstrated use of power wasters to cause a DoS on both the CPU and the FPGA in an SoC platform [73] strongly suggests that it may be possible to inject faults across different components. Furthermore, the demonstrated side-channel attacks [77] show that the power network is tightly coupled, confirming that power-related effects across components are possible. Power-wasting circuits can be even more powerful if combined with DVFS or CPU stressor codes to inject faults, while bypassing the countermeasures that focus on either of the two threats alone. Finally, the recent demonstration of using clock gating to boost the FPGA transient power consumption [26] shows that clock gating, and potentially power gating, may be used in the attacks. Knowing that powering on large clusters of gated logic may cause current surges [203], it becomes important to investigate whether power gating the islands in an MPSoC may facilitate some of the exploits.

Software-based fault-injection attacks on heterogeneous systems require new threat models. For instance, access to the DVFS interfaces or their manipulation beyond specified OPPs may no longer be required given that the adversary can program the FPGA to inject faults that are not localized to the programmable fabric. The proposed and deployed countermeasures cannot protect all of the computing components against all of the fault injection possibilities. However, this does not necessarily mean that it should be impossible to protect a heterogeneous system against fault-injection attacks. Combining careful design choices with countermeasures at the algorithmic level [16], the software drivers and hardware [112] may enable the detection of the fault injection before it can be exploited. Furthermore, restricting the access to the exploitable software and hardware interfaces or ensuring that they cannot be used to operate the circuit beyond its functional limits would render some interfaces safe and, consequently, software-based exploits more difficult. Therefore, new research is required to find and tailor the countermeasures for heterogeneous systems.

8 CONCLUSION

In this work, we have surveyed the electrical-level attacks on CPUs, FPGAs, and GPUs, which are the main components of today's heterogeneous systems. While previous work considered the security of each computing unit on its own, we paid special attention to the potential security threats brought about by their integration. Considering the architecture of common heterogeneous systems, we investigated whether electrical-level attacks may pose a threat to a heterogeneous platform as a whole. Finally, we proposed a number of future research directions that, if addressed, should help to ensure the security of today's heterogeneous computing systems.

REFERENCES

[1] Anita Aghaie, Amir Moradi, Shahram Rasoolzadeh, Aein Rezaei Shahmirzadi, Falk Schellenberg, and Tobias Schneider. 2020. Impeccable circuits. *Transactions on Computers* 69, 3 (Mar 2020), 361–376.

[2] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. 2010. When clocks fail: On critical paths and clock faults. In *International Conference on Smart Card Research and Advanced Applications*. Springer, Passau, Germany, 182–93.

[3] Ibrahim Ahmed, Linda L. Shen, and Vaughn Betz. 2020. Optimizing FPGA logic circuitry for variable voltage supplies. *IEEE Trans. on VLSI* 28, 4 (April 2020), 890–903.

[4] Gökhan Akgün, Lester Kalms, and Diana Göhringer. 2020. Resource efficient dynamic voltage and frequency scaling on Xilinx FPGAs. In *Applied Reconfigurable Computing. Architectures, Tools, and Applications*. Springer, Toledo, Spain, 178–192.

[5] Md Mahbub Alam, Shahin Tajik, Fatemeh Ganji, Mark Tehranipoor, and Domenic Forte. 2019. RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Atlanta, GA, 48–55.

[6] Alibaba [n.d.]. Compute optimized instance families with FPGAs - Instance| Alibaba Cloud Documentation Center. Retrieved December 2, 2021 from https://www.alibabacloud.com/help/doc-detail/108504.htm.

[7] Amazon F1 [n.d.]. FPGA-Based Amazon EC2 F1 Computing Instances. Retrieved December 2, 2021 from aws.amazon.com/ec2/instance-types/f1/.

[8] Arria10. 2021. Intel Arria 10 Hard Processor System Technical Reference Manual. Retrieved December 2, 2021 from intel.com.

[9] Christian Aumüller, Peter Bier, Wieland Fischer, Peter Hofreiter, and Jean-Pierre Seifert. 2002. Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In *Cryptographic Hardware and Embedded Systems*. Springer, Redwood Shores, CA, 260–275.

[10] Zelalem Birhanu Aweke, Salessawi Ferede Yitbarek, Rui Qiao, Reetuparna Das, Matthew Hicks, Yossi Oren, and Todd Austin. 2016. ANVIL: Software-based protection against next-generation RowHammer attacks. In *International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, Atlanta, GA, 743–755.

[11] Furkan Aydin, Priyank Kashyap, Seetal Potluri, Paul Franzon, and Aydin Aysu. 2020. DeePar-SCA: Breaking parallel architectures of lattice cryptography via learning based side-channel attacks. In *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer, Samos, Greece, 262–280.

[12] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. 2011. An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Nara, Japan, 105–114.

[13] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. 2006. The sorcerer's apprentice guide to fault attacks. *Proc. IEEE* 94, 2 (Feb. 2006), 370–382.

[14] Alessandro Barenghi, Guido M. Bertoni, Luca Breveglieri, and Gerardo Pelosi. 2013. A fault induction technique based on voltage underfeeding with application to attacks against AES and RSA. *Journal of Systems and Software* 86, 7 (July 2013), 1864–1878.

[15] Alessandro Barenghi, Luca Breveglieri, Israel Koren, and David Naccache. 2012. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proc. IEEE* 100, 11 (Nov 2012), 3056–3076.

[16] Alessandro Barenghi, Luca Breveglieri, Israel Koren, Gerardo Pelosi, and Francesco Regazzoni. 2010. Countermeasures against fault attacks on software implemented AES: Effectiveness and cost. In *5th Workshop on Embedded Systems Security*. ACM, Scottsdale, AZ, 1–10.

[17] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *Security Symposium*. USENIX, Santa Clara, CA, 515–532.

[18] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. Poster: Recovering the input of neural networks via single shot side-channel attacks. In *Conference on Computer and Communications Security*. ACM, London, United Kingdom, 2657–2659.

[19] Ali Galip Bayrak, Francesco Regazzoni, David Novo, Philip Brisk, François-Xavier Standaert, and Paolo Ienne. 2015. Automatic application of power analysis countermeasures. *IEEE Trans. Comput.* 64, 2 (Feb 2015), 329–341.

[20] Noura Benhadjyoussef, Hassen Mestiri, Mohsen Machhout, and Rached Tourki. 2012. Implementation of CPA analysis against AES design on FPGA. In *International Conference on Communications and Information Technology (ICCIT'12)*. IEEE, Hammamet, Tunisia, 124–128.

[21] Shivam Bhasin, Nidhal Selmane, Sylvain Guilley, and Jean-Luc Danger. 2009. Security evaluation of different AES implementations against practical setup time violation attacks in FPGAs. In *International Workshop on Hardware-Oriented Security and Trust*. IEEE, San Francisco, CA, 15–21.

[22] Alex Biryukov, Daniel Dinu, and Johann Großschädl. 2016. Correlation power analysis of lightweight block ciphers: From theory to practice. In *International Conference on Applied Cryptography and Network Security*. Springer, Guildford, UK, 537–557.

[23] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. 1997. On the importance of checking cryptographic protocols for faults. In *Advances in Cryptology—EUROCRYPT'97*. Springer, Konstanz, Germany, 37–51.

[24] Talal Bonny and Qassim Nasir. 2019. Clock glitch fault injection attack on an FPGA-based non-autonomous chaotic oscillator. *Nonlinear Dynamics* 96, 3 (May 2019), 2087–2101.

[25] Erik Bosman, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. 2016. Dedup est machina: Memory deduplication as an advanced exploitation vector. In *IEEE Symposium on Security and Privacy (S&P'16)*. IEEE, San Jose, CA, 987–1004.

[26] Andrew Boutros, Mathew Hall, Nicolas Papernot, and Vaughn Betz. 2020. Neighbors from hell: Voltage attacks against deep learning accelerators on multi-tenant FPGAs. In *IEEE International Conference on Field Programmable Technology*. IEEE, Maui, HI, 9 pages.

[27] Claudio Bozzato, Riccardo Focardi, and Francesco Palmarini. 2019. Shaping the glitch: Optimizing voltage fault injection attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019, 2 (Feb 2019), 199–224.

[28] Ferdinand Brasser, Lucas Davi, David Gens, Christopher Liebchen, and Ahmad-Reza Sadeghi. 2017. CAn't Touch This: Software-only mitigation against RowHammer attacks targeting kernel memory. In *26th Security Symposium*. USENIX, Vancouver, BC, 117–130.

[29] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems*. Springer, Cambridge, MA, 16–29.

[30] Robert Buhren, Hans Niklas Jacob, Thilo Krachenfels, and Jean-Pierre Seifert. 2021. One Glitch to Rule Them All: Fault Injection Attacks Against AMD's Secure Encrypted Virtualization. Retrieved December 2, 2021 from arXiv:2108.04575 [cs.CR]

[31] Jo Van Bulck, Frank Piessens, and Raoul Strackx. 2017. SGX-step: A practical attack framework for precise enclave execution control. In *2nd Workshop on System Software for Trusted Execution*. ACM, Shanghai, China, 1–6.

[32] Henri Calandra, Romain Dolbeau, Pierre Fortin, Jean-Luc Lamotte, and Issam Said. 2013. Evaluation of successive CPUs/APUs/GPUs based on an openCL finite difference stencil. In *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, Belfast, UK, 405–409.

[33] Gaetan Canivet, Paolo Maistri, Regis Leveugle, Jessy Clédière, Frederic Valette, and Marc Renaudin. 2011. Glitch and laser fault attacks onto a secure AES implementation on a SRAM-based FPGA. *Journal of Cryptology* 24, 2 (Apr 2011), 247–268.

[34] Vincent Carlier, Herve Chabanne, Emmanuelle Dottax, and Herve Pelletier. 2005. Generalizing square attack using side-channels of an AES implementation on an FPGA. In *15th International Conference on Field-Programmable Logic and Applications*. IEEE, Tampere, Finalnd, 433–437.

[35] Rafael Boix Carpi, Stjepan Picek, Lejla Batina, Federico Menarini, Domagoj Jakobovic, and Marin Golub. 2013. Glitch it if you can: Parameter search strategies for successful fault injection. In *International Conference on Smart Card Research and Advanced Applications*. Springer, Berlin, Germany, 236–252.

[36] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. 2002. Template attacks. In *Cryptographic Hardware and Embedded Systems*. Springer, Redwood Shores, CA, 13–28.

[37] Zitai Chen, Georgios Vasilakis, Kit Murdock, Edward Dean, David Oswald, and Flavio D. Garcia. 2021. VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface. In *30th Security Symposium*. USENIX, Vancouver, BC, 1–18.

[38] Young-Kyu Choi, Jason Cong, Zhenman Fang, Yuchen Hao, Glenn Reinman, and Peng Wei. 2019. In-depth analysis on microarchitectures of modern heterogeneous CPU-FPGA platforms. *ACM Trans. on Reconfigurable Technology and Systems* 12, 1 (Apr 2019), 1–20.

[39] Christophe Clavier. 2004. Side Channel Analysis for Reverse Engineering (SCARE) — An Improved Attack Against a Secret A3/A8 GSM Algorithm. Cryptology ePrint Archive, Report 2004/049, 14 pages. https://eprint.iacr.org/2004/049.

[40] Christophe Clavier, Benoit Feix, Georges Gagnerot, Myléne Roussellet, and Vincent Verneuil. 2010. Horizontal correlation analysis on exponentiation. In *International Conference on Information and Communications Security*. Springer, Barcelona, Spain, 46–61.

[41] Lucian Cojocar, Jeremie Kim, Minesh Patel, Lillian Tsai, Stefan Saroiu, Alec Wolman, and Onur Mutlu. 2020. Are we susceptible to RowHammer? An end-to-end methodology for cloud providers. In *IEEE Symposium on Security and Privacy (S&P'20)*. IEEE, San Francisco, CA, 712–728.

[42] Lucian Cojocar, Kostas Papagiannopoulos, and Niek Timmers. 2017. Instruction duplication: Leaky and not too fault-tolerant!. In *International Conference on Smart Card Research and Advanced Applications*. Springer, Lugano, Switzerland, 160–179.

[43] Patrick Cronin, Xing Gao, Chengmo Yang, and Haining Wang. 2021. Charger-surfing: Exploiting a power line side-channel for smartphone information leakage. In *30th USENIX Security Symposium*. USENIX, Vancouver, BC, 1–18.

[44] Ang Cui and Rick Housley. 2017. BADFET: Defeating modern secure boot using second-order pulsed electromagnetic fault injection. In *11th USENIX Workshop on Offensive Technologies (WOOT'17)*. USENIX, Vancouver, BC, 1–12.

[45] Cyclone V. 2020. Cyclone V Hard Processor System Technical Reference Manual. Retrieved December 2, 2021 from intel.com.

[46] Amine Dehbaoui, J. Dutertre, Bruno Robisson, and Assia Tria. 2012. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Leuven, Belgium, 7–15.

[47] Amine Dehbaoui, Amir-Pasha Mirbaha, Nicolas Moro, Jean-Max Dutertre, and Assia Tria. 2013. Electromagnetic glitch on the AES round counter. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, Paris, France, 17–31.

[48] José Duato, Antonio J. Peña, Federico Silla, Rafael Mayo, and Enrique S. Quintana-Ortí. 2010. rCUDA: Reducing the number of GPU-based accelerators in high performance clusters. In *2010 International Conference on High Performance Computing Simulation*. IEEE, Caen, France, 224–231.

[49] Anuj Dubey, Rosario Cammarota, and A. Aysu. 2020. MaskedNet: The first hardware inference engine aiming power side-channel protection. In *International Symposium on Hardware-Oriented Security and Trust*. IEEE, San Jose, CA, 197–208.

[50] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. 2010. *Building a Side Channel Based Disassembler*. Springer, Berlin, 78–99.

[51] Ted Eisenberg, David Gries, Juris Hartmanis, Don Holcomb, M. Stuart Lynn, and Thomas Santoro. 1989. The Cornell Commission: On Morris and the worm. *Commun. ACM* 32, 6 (Jun 1989), 706–709.

[52] Ilija Ekmečić, Igot Tartalja, and Veljko Milutinović. 1996. A survey of heterogeneous computing: Concepts and systems. *Proc. IEEE* 84, 8 (Aug 1996), 1127–1144.

[53] M. A. Elmohr, H. Liao, and C. H. Gebotys. 2020. EM fault injection on ARM and RISC-V. In *21st International Symposium on Quality Electronic Design (ISQED'20)*. IEEE, Santa Clara, CA, 206–212.

[54] Rana Elnaggar, Siyuan Chen, Peilin Song, and Krishnendu Chakrabarty. 2020. Detection of RowHammer attacks in SoCs with FPGAs. In *IEEE European Test Symposium (ETS'20)*. IEEE, Tallinn, Estonia, 1–2.

[55] Dan Ernst, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham, Conrad Ziesler, David Blaauw, Todd Austin, Krisztián Flautner, and Trevor Mudge. 2003. Razor: A low-power pipeline based on circuit-level timing speculation. In *36th International Symposium on Microarchitecture (MICRO-36)*. IEEE, San Diego, CA, USA, 7–18.

[56] Muhammad Arsath K. F, Vinod Ganesan, Rahul Bodduna, and Chester Rebeiro. 2020. PARAM: A microprocessor hardened for power side-channel attack resistance. In *International Symposium on Hardware-Oriented Security and Trust*. IEEE, San Jose, CA, USA, 23–34.

[57] Michael S. Floyd, Phillip J. Restle, Michael A. Sperling, Pawel Owczarczyk, Eric J. Fluhr, Joshua Friedrich, Paul Muench, Timothy Diemoz, Pierce Chuang, and Christos Vezyrtzis. 2017. Adaptive clocking in the POWER9™ Processor for voltage droop protection. In *International Solid-State Circuits Conference*. IEEE, San Francisco, CA, 444–45.

[58] Pietro Frigo, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Grand pwning unit: Accelerating microarchitectural attacks with the GPU. In *IEEE Symposium on Security and Privacy (S&P'18)*. IEEE, San Francisco, CA, 195–210.

[59] Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2020. TRRespass: Exploiting the many sides of target row refresh. In *IEEE Symposium on Security and Privacy (S&P'20)*. IEEE, San Francisco, CA, 747–762.

[60] Martin Gag, Tim Wegner, Ansgar Waschki, and Dirk Timmermann. 2012. Temperature and on-chip crosstalk measurement using ring oscillators in FPGA. In *15th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS'12)*. IEEE, Tallinn, Estonia, 201–204.

[61] Yiwen Gao and Yongbin Zhou. 2020. Side-channel attacks with multi-thread mixed leakage. *IEEE Trans. Inf. Forensics Secur.* 16 (Sept. 2020), 770–785.

[62] Yiwen Gao, Yongbin Zhou, and Wei Cheng. 2020. Efficient electro-magnetic analysis of a GPU bitsliced AES implementation. *Cybersecurity* 3, 1 (Feb 2020), 17 pages.

[63] Rong Ge, Ryan Vogt, Jahangir Majumder, Arif Alam, Martin Burtscher, and Ziliang Zong. 2013. Effects of dynamic voltage and frequency scaling on a K20 GPU. In *42nd International Conference on Parallel Processing*. IEEE, Lyon, France, 826–833.

[64] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. 2015. Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *Cryptographic Hardware and Embedded Systems*. Springer, Saint-Malo, France, 207–228.

[65] Daniel Genkin, Itamar Pipman, and Eran Tromer. 2014. Get your hands off my laptop: Physical side-channel key-extraction attacks on PCs. In *Cryptographic Hardware and Embedded Systems*. Springer, Busan, Korea, 242–260.

[66] Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa Taha, and Patrick Schaumont. 2014. Differential fault intensity analysis. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Busan, South Korea, 49–58.

[67] Ilias Giechaskiel, Kaspar B. Rasmussen, and Jakub Szefer. 2019. Measuring long wire leakage with ring oscillators in cloud FPGAs. In *29th International Conference on Field-Programmable Logic and Applications*. IEEE, Barcelona, Spain, 45–50.

[68] Ilias Giechaskiel, Kaspar B. Rasmussen, and Jakub Szefer. 2020. C3APSULe: Cross-FPGA covert-channel attacks through power supply unit leakage. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, San Francisco, CA, 1728–1741.

[69] Ilias Giechaskiel and Jakub Szefer. 2020. Information leakage from FPGA routing and logic elements. In *IEEE/ACM International Conference on Computer-Aided Design*. ACM, Virtual, 1–9.

[70] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. 2008. Mutual information analysis. In *Cryptographic Hardware and Embedded Systems*. Springer, Washington, DC, 426–442.

[71] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. 2020. Are cloud FPGAs really vulnerable to power analysis attacks?. In *Design, Automation and Test in Europe*. IEEE, Grenoble, France, 1–4.

[72] Dennis R. E. Gnad, Vincent Meyers, Nguyen Minh Dang, Falk Schellenberg, Amir Moradi, and Mehdi B. Tahoori. 2021. Stealthy logic misuse for power analysis attacks in multi-tenant FPGAs. In *Design, Automation and Test in Europe*. IEEE, Virtual, 4 pages.

[73] Dennis R. E. Gnad, Fabian Oboril, and Mehdi B. Tahoori. 2017. Voltage drop-based fault attacks on FPGAs using valid bitstreams. In *27th International Conference on Field-Programmable Logic and Applications*. IEEE, Ghent, Belgium, 1–7.

[74] Cheng Gongye, Hongjia Li, Xiang Zhang, Majid Sabbagh, Geng Yuan, Xue Lin, Thomas Wahl, and Yunsi Fei. 2020. New passive and active attacks on deep neural networks in medical applications. In *IEEE/ACM International Conference on Computer-Aided Design*. ACM, Virtual, 1–9.

[75] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, and Philippe Loubet-Moundi. 2019. High-speed ring oscillator based sensors for remote side-channel attacks on FPGAs. In *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, Cancun, Mexico, 1–8.

[76] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, and Philippe Loubet Moundi. 2020. SideLine: How Delay-Lines (May) Leak Secrets from your SoC. Retrieved December 2, 2021 from arXiv:2009.07773 [cs.CR]

[77] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, Philippe Loubet Moundi, and Francis Olivier. 2019. Remote side-channel attacks on heterogeneous SoC. In *International Conference on Smart Card Research and Advanced Applications*. Springer, Prague, Czech Republic, 109–125.

[78] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varıcı. 2014. LS-Designs: Bitslice encryption for efficient masked software implementations. In *Conference on Fast Software Encryption*. Springer, London, UK, 18–37.

[79] Johann Großschädl, Elisabeth Oswald, Dan Page, and Michael Tunstall. 2009. Side-channel analysis of cryptographic software via early-terminating multiplications. In *International Conference on Information, Security and Cryptology*. Springer, Seoul, Korea, 176–192.

[80] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2016. Rowhammer.js: A remote software-induced fault attack in JavaScript. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, San Sebastián, Spain, 300–321.

[81] Tim Güneysu and Amir Moradi. 2011. Generic side-channel countermeasures for reconfigurable devices. In *Cryptographic Hardware and Embedded Systems*. Springer, Nara, Japan, 33–48.

[82] Zacharias Hadjilambrou, Shidhartha Das, Marco A. Antoniades, and Yiannakis Sazeides. 2021. Harnessing CPU electromagnetic emanations for resonance-induced voltage-noise characterization. *IEEE Trans. Comput.* 70, 9 (Sept. 2021), 1338–1349.

[83] Yi Han, Ioannis Christoudis, Konstantinos I. Diamantaras, Saman Zonouz, and Athina Petropulu. 2019. Side-channel-based code-execution monitoring systems: A survey. *IEEE Signal Processing Magazine* 36, 2 (Mar 2019), 22–35.

[84] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. 2020. Applications of machine learning techniques in side-channel attacks: A survey. *Journal of Cryptographic Engineering* 10, 2 (Jun 2020), 135–162.

[85] Annelie Heuser and Michael Zohner. 2012. Intelligent machine homicide. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, Darmstadt, Germany, 249–264.

[86] Johann Heyszl, Dominik Merli, Benedikt Heinz, Fabrizio De Santis, and Georg Sigl. 2012. Strengths and limitations of high-resolution electromagnetic field measurements for side-channel analysis. In *International Conference on Smart Card Research and Advanced Applications*. Springer, Graz, Austria, 248–262.

[87] Sanghyun Hong, Pietro Frigo, Yiğitcan Kaya, Cristiano Giuffrida, and Tudor Dumitraş. 2019. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *28th Security Symposium*. USENIX, Santa Clara, CA, 497–514.

[88] Yohei Hori, Toshihiro Katashita, Akihiko Sasaki, and Akashi Satoh. 2012. Electromagnetic side-channel attack against 28-nm FPGA device. In *Pre-proceedings of WISA*. Jeju, Korea, 9 pages.

[89] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. 2011. Machine learning in side-channel analysis: A first study. *Journal of Cryptographic Engineering* 1, 4 (Dec 2011), 293–302.

[90] HSA [n.d.]. HSA Specification Library. Retrieved December 2, 2021 from http://www.hsafoundation.com/html_spec111/HSA_Library.htm.

[91] Huawei [n.d.]. FPGA Accelerated Cloud Server-Huawei Cloud. Retrieved December 2, 2021 from https://www.huaweicloud.com/en-us/product/fcs.html.

[92] Ted Huffmire, Brett Brotherton, Nick Callegari, Jonathan Valamehr, Jeff White, Ryan Kastner, and Tim Sherwood. 2008. Designing secure systems on reconfigurable hardware. *Trans. on Design Automation of Electronic Systems* 13, 3 (July 2008), 44:1–44:24.

[93] Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. 2018. MASCAT: Preventing microarchitectural attacks before distribution. In *Conference on Data and Application Security and Privacy*. ACM, Tempe, AZ, 377–88.

[94] Yeongjin Jang, Jaehyuk Lee, Sangho Lee, and Taesoo Kim. 2017. SGX-Bomb: Locking down the processor via RowHammer attack. In *2nd Workshop on System Software for Trusted Execution*. ACM, Shanghai, China, 1–6.

[95] Weixiong Jiang, Rui Li, Heng Yu, and Yajun Ha. 2020. An accurate FPGA online delay monitor supporting all timing paths. In *International Symposium on Circuits and Systems (ISCAS'20)*. IEEE, Seville, Spain, 1–5.

[96] Zhen Hang Jiang, Yunsi Fei, and David Kaeli. 2016. A complete key recovery timing attack on a GPU. In *International Symposium on High Performance Computer Architecture (HPCA'16)*. IEEE, Barcelona, Spain, 394–405.

[97] Duško Karaklajić, Jörn-Marc Schmidt, and Ingrid Verbauwhede. 2013. Hardware designer's guide to fault attacks. *IEEE Trans. on VLSI* 21, 12 (Dec 2013), 2295–306.

[98] Zahra Kazemi, Athanasios Papadimitriou, Ioanna Souvatzoglou, Ehsan Aerabi, Mosabbah Mushir Ahmed, David Hely, and Vincent Beroulle. 2019. On a low cost fault injection framework for security assessment of cyber-physical systems: Clock glitch attacks. In *International Verification and Security Workshop*. IEEE, Rhodes, Greece, 7–12.

[99] Zijo Kenjar, Tommaso Frassetto, David Gens, Michael Franz, and Ahmad-Reza Sadeghi. 2020. V0LTpwn: Attacking x86 processor integrity from software. In *29th Security Symposium*. USENIX, Virtual, 1445–1461.

[100] ChangKyun Kim, Martin Schläffer, and SangJae Moon. 2008. Differential side channel analysis attacks on FPGA implementations of ARIA. *ETRI Journal* 30, 2 (Apr 2008), 315–325.

[101] Jeremie S. Kim, Minesh Patel, A. Giray Yağlıkçı, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu. 2020. Revisiting RowHammer: An experimental analysis of modern DRAM devices and mitigation techniques. In *47th Annual International Symposium on Computer Architecture (ISCA'20)*. IEEE, Valencia, Spain, 638–651.

[102] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *41st Annual International Symposium on Computer Architecture*. IEEE, Minneapolis, MN, 361–372.

[103] Youngtaek Kim and Lizy Kurian John. 2011. Automated di/dt stressmark generation for microprocessor power delivery networks. In *International Symposium on Low Power Electronics and Design*. IEEE, Fukuoka, Japan, 253–258.

[104] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. 2019. Spectre attacks: Exploiting speculative execution. In *IEEE Symposium on Security and Privacy (S&P'19)*. IEEE, San Francisco, CA, 1–19.

[105] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Advances in Cryptology—CRYPTO'99*. Springer, Santa Barbara, CA, 388–97.

[106] Radhesh Krishnan Konoth, Marco Oliverio, Andrei Tatar, Dennis Andriesse, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. 2018. ZebRAM: Comprehensive and compatible software protection against RowHammer attacks. In *13th USENIX Symposium on Operating Systems Design and Implementation*. USENIX, Carlsbad, CA, 697–710.

[107] Thomas Korak and Michael Hoefler. 2014. On the effects of clock and power supply tampering on two microcontroller platforms. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Busan, South Korea, 8–17.

[108] Thomas Korak, Michael Hutter, Baris Ege, and Lejla Batina. 2014. Clock glitch attacks in the presence of heating. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Busan, South Korea, 104–114.

[109] Jakub Korczyc and Andrzej Krasniewski. 2012. Evaluation of susceptibility of FPGA-based circuits to fault injection attacks based on clock glitching. In *15th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS'12)*. IEEE, Talinn, Estonia, 171–174.

[110] Jonas Krautter, Dennis R. E. Gnad, Falk Schellenberg, Amir Moradi, and Mehdi B. Tahoori. 2019. Active fences against voltage-based side channels in multi-tenant FPGAs. In *IEEE/ACM International Conference on Computer-Aided Design*. ACM, Westminster, CO, 1–8.

[111] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. 2018. FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018, 3 (Aug. 2018), 44–68.

[112] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. 2019. Mitigating electrical-level attacks towards secure multi-tenant FPGAs in the cloud. *ACM Trans. on Reconfigurable Technology and Systems* 12, 3 (Aug. 2019), 1–26.

[113] Deepak Krishnankutty, Zheng Li, Ryan Robucci, Nilanjan Banerjee, and Chintan Patel. 2020. Instruction sequence identification and disassembly using power supply side-channel analysis. *IEEE Trans. Comput.* 69, 11 (Nov. 2020), 1639–1653.

[114] Sebastian Kutzner, Axel Poschmann, and Marc Stöttinger. 2013. TROJANUS: An ultra-lightweight side-channel leakage generator for FPGAs. In *IEEE International Conference on Field Programmable Technology*. IEEE, Kyoto, Japan, 160–167.

[115] Tuan Minh La, Kaspar Matas, Nikola Grunchevski, Khoa Dang Pham, and Dirk Koch. 2020. FPGADefender: Malicious self-oscillator scanning for Xilinx ultrascale + FPGAs. *TRETS* 13, 3, Article 15 (Sept. 2020), 31 pages.

[116] Eojin Lee, Ingab Kang, Sukhan Lee, G. Edward Suh, and Jung Ho Ahn. 2019. TWiCe: Preventing row-hammering by exploiting time window counters. In *46th International Symposium on Computer Architecture*. ACM, Phoenix, AZ, 385–396.

[117] Lang Lin, Markus Kasper, Tim Güneysu, Christof Paar, and Wayne Burleson. 2009. Trojan side-channels: Lightweight hardware Trojans through side-channel engineering. In *Cryptographic Hardware and Embedded Systems*. Springer, Lausanne, Switzerland, 382–395.

[118] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-based power side-channel attacks on x86. In *IEEE Symposium on Security and Privacy (S&P'21)*. IEEE, Virtual, 355–371.

[119] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, and et al. 2018. Meltdown: Reading kernel memory from user space. In *27th Security Symposium*. USENIX, Baltimore, MD, 973–90.

[120] Moritz Lipp, Michael Schwarz, Lukas Raab, Lukas Lamster, Misiker Tadesse Aga, Clémentine Maurice, and Daniel Gruss. 2020. Nethammer: Inducing RowhHammer faults through network requests. In *European Symposium on Security and Privacy Workshops*. IEEE, Genoa, Italy, 710–719.

[121] Hailong Liu, Zhenglin Liu, Yifei Qiao, and Zhaojun Lu. 2017. Clock glitch fault injection attacks on an FPGA AES implementation. *Journal of Electrotechnology, Electrical Engineering and Management* 1, 1 (Mar 2017), 23–27.

[122] Wenye Liu, Chip-Hong Chang, and Fan Zhang. 2021. Stealthy and robust glitch injection attack on deep learning accelerator for target with variational viewpoint. *IEEE Trans. Inf. Forensics Secur.* 16 (2021), 1928–42.

[123] Jake Longo, Elke De Mulder, Daniel Page, and Michael Tunstall. 2015. SoC It to EM: Electromagnetic side-channel attacks on a complex system-on-chip. In *Cryptographic Hardware and Embedded Systems*. Springer, Saint-Malo, France, 620–640.

[124] Chao Luo, Yunsi Fei, Pei Luo, Saoni Mukherjee, and David Kaeli. 2015. Side-channel power analysis of a GPU AES implementation. In *33rd IEEE International Conference on Computer Design (ICCD'15)*. IEEE, New York, NY, 281–288.

[125] Chao Luo, Yunsi Fei, Liwei Zhang, A. Adam Ding, Pei Luo, Saoni Mukherjee, and David Kaeli. 2018. Power analysis attack of an AES GPU implementation. *Journal of Hardware and Systems Security* 2, 1 (Mar 2018), 69–82.

[126] Pei Luo, Chao Luo, and Yunsi Fei. 2016. System Clock and Power Supply Cross-Checking for Glitch Detection. Cryptology ePrint Archive, Report 2016/968. https://eprint.iacr.org/2016/968.

[127] Yukui Luo and Xiaolin Xu. 2019. HILL: A hardware isolation framework against information leakage on multi-tenant FPGA long-wires. In *IEEE International Conference on Field Programmable Technology*. IEEE, Tianjin, China, 12–19.

[128] Yukui Luo and Xiaolin Xu. 2020. A quantitative defense framework against power attacks on multi-tenant FPGA. In *IEEE/ACM International Conference on Computer-Aided Design*. IEEE, San Diego, CA, 1–4.

[129] Maxime Madau, Michel Agoyan, Josep Balasch, Miloš Grujić, Patrick Haddad, Philippe Maurine, Vladimir Rožić, Dave Singelée, Bohan Yang, and Ingrid Verbauwhede. 2018. The impact of pulsed electromagnetic fault injection on true random number generators. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Amsterdam, Netherlands, 43–48.

[130] Dina Mahmoud and Mirjana Stojilović. 2019. Timing violation induced faults in multi-tenant FPGAs. In *Design, Automation and Test in Europe*. IEEE, Florence, Italy, 1745–1750.

[131] Dina G. Mahmoud, Wei Hu, and Mirjana Stojilović. 2020. X-Attack: Remote activation of satisfiability don't-care hardware Trojans on shared FPGAs. In *30th International Conference on Field-Programmable Logic and Applications*. IEEE, Gothenburg, Sweden, 185–192.

[132] Antun Maldini, Niels Samwel, Stjepan Picek, and Lejla Batina. 2019. *Optimizing Electromagnetic Fault Injection with Genetic Algorithms*. Springer, 281–300.

[133] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2007. *Power Analysis Attacks—Revealing the Secrets of Smart Cards*. Springer, New York, NY.

[134] Heiko Mantel, Johannes Schickel, Alexandra Weber, and Friedrich Weber. 2018. How secure is green IT? The case of software-based energy side channels. In *23rd European Symposium on Research in Computer Security*. Springer, Barcelona, Spain, 218–239.

[135] Adrien Le Masle, Gary C. T. Chow, and Wayne Luk. 2011. Constant power reconfigurable computing. In *IEEE International Conference on Field Programmable Technology*. IEEE, New Delhi, India, 1–8.

[136] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. 2020. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 1 (2020), 348–375.

[137] Kaspar Matas, Tuan Minh La, Khoa Dang Pham, and Dirk Koch. 2020. Power-hammering through glitch amplification—attacks and mitigation. In *28th Symposium on Field-Programmable Custom Computing Machines*. IEEE, Fayetteville, AR, 65–69.

[138] Masato Matsubayashi, Akashi Satoh, and Jun Ishii. 2016. Clock glitch generator on SAKURA-G for fault injection attack against a cryptographic circuit. In *The 5th Global Conference on Consumer Electronics*. IEEE, Kyoto, Japan, 1–4.

[139] David McCann, Elisabeth Oswald, and Carolyn Whitnall. 2017. Towards practical tools for side channel aware software engineering: 'grey box' modelling for instruction leakages. In *26th Security Symposium*. USENIX, Vancouver, BC, 199–216.

[140] Steven McNeil, Peter Schillinger, Aniket Kolarkar, Emmanuel Puillet, and Uwe Gertheinrich. 2020. Isolation Methods in Zynq UltraScale+ MPSoCs. Xilinx Application Note, XAPP132.

[141] Alexandre Menu, Shivam Bhasin, Jean-Max Dutertre, Jean-Baptiste Rigaud, and Jean-Luc Danger. 2019. Precise spatio-temporal electromagnetic fault injections on data transfers. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Atlanta, GA, 1–8.

[142] Microsoft Azure [n.d.]. Machine Learning. Retrieved December 2, 2021 from https://azure.microsoft.com/en-us/pricing/details/machine-learning/.

[143] Seyedeh Sharareh Mirzargar, Gaiëtan Renault, Andrea Guerrieri, and Mirjana Stojilović. 2020. Nonintrusive and adaptive monitoring for locating voltage attacks in virtualized FPGAs. In *IEEE International Conference on Field Programmable Technology*. IEEE, Maui, HI, 1–2.

[144] Seyedeh Sharareh Mirzargar and Mirjana Stojilović. 2019. Physical side-channel attacks and covert communication on FPGAs: A survey. In *29th International Conference on Field-Programmable Logic and Applications*. IEEE, Barcelona, Spain, 202–210.

[145] Noriyuki Miura, Zakaria Najm, Wei He, Shivam Bhasin, Xuan Thuy Ngo, Makoto Nagata, and Jean-Luc Danger. 2016. PLL to the rescue: A novel EM fault countermeasure. In *53rd Annual Design Automation Conference*. IEEE, Austin, TX, 1–6.

[146] Shayan Moini, Shanquan Tian, Jakub Szefer, Daniel Holcomb, and Russell Tessier. 2020. Remote Power Side-Channel Attacks on CNN Accelerators in FPGAs. Retrieved December 2, 2021 from arXiv: 2011.07603.

[147] Amir Moradi. 2014. Side-channel leakage through static power. In *Cryptographic Hardware and Embedded Systems*. Springer, Busan, South Korea, 562–579.

[148] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. 2011. On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs. In *Conference on Computer and Communications Security*. ACM, Chicago, IL, 111–124.

[149] Amir Moradi, David Oswald, Christof Paar, and Pawel Swierczynski. 2013. Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: Facilitating black-box analysis using software reverse-engineering. In *21st ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM, Monterey, CA, 91–100.

[150] Amir Moradi and Tobias Schneider. 2016. Improved side-channel analysis attacks on Xilinx bitstream encryption of 5, 6, and 7 series. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, Graz, Austria, 71–87.

[151] Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. 2013. Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Los Alamitos, CA, 77–88.

[152] Nicolas Moro, Karine Heydemann, Amine Dehbaoui, Bruno Robisson, and Emmanuelle Encrenaz. 2014. Experimental evaluation of two software countermeasures against fault attacks. In *International Symposium on Hardware-Oriented Security and Trust*. IEEE, Arlington, VA, 112–117.

[153] Saoni Mukherjee, Yunsi Fei, and David Kaeli. 2019. Poster: GIPSim: Low-level power modeling for resiliency from side channel attacks on GPUs. In *IEEE Symposium on Security and Privacy (S&P'19)*. IEEE, San Francisco, CA, 2 pages.

[154] Naila Mukhtar, Mohamad Ali Mehrabi, Yinan Kong, and Ashiq Anjum. 2019. Machine-learning-based side-channel evaluation of elliptic-curve cryptographic FPGA processor. *Applied Sciences* 9, 11 (Jan 2019), 64–84.

[155] Elke De Mulder, P. Buysschaert, Sıddıka Berna Örs, P. Delmotte, Bart Preneel, Guy Vandenbosch, and Ingrid Verbauwhede. 2005. Electromagnetic analysis attack on an FPGA implementation of an elliptic curve cryptosystem. In *EUROCON 2005 — The International Conference on "Computer as a Tool"*, Vol. 2. IEEE, Belgrade, Serbia, 1879–1882.

[156] Elke De Mulder, Samatha Gummalla, and Michael Hutter. 2019. Protecting RISC-V against side-channel attacks. In *56th ACM/ESDA/IEEE Design Automation Conference*. ACM, Las Vegas, NV, 1–4.

[157] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. 2020. Plundervolt: Software-based fault injection attacks against Intel SGX. In *IEEE Symposium on Security and Privacy (S&P'20)*. IEEE, San Francisco, CA, 1466–1482.

[158] Onur Mutlu and Jeremie S. Kim. 2019. RowHammer: A retrospective. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 39, 8 (May 2019), 1555–1571.

[159] Safouane Noubir, Maria Mendez Real, and Sebastien Pillement. 2020. Towards malicious exploitation of energy management mechanisms. In *Design, Automation and Test in Europe*. IEEE, Grenoble, France, 1043–1048.

[160] Johannes Obermaier, Robert Specht, and Georg Sigl. 2017. Fuzzy-glitch: A practical ring oscillator based clock glitch attack. In *International Conference on Applied Electronics (AE'17)*. IEEE, Pilsen, Czech Republic, 1–6.

[161] Sebastien Ordas, Ludovic Guillaume-Sage, and Philippe Maurine. 2017. Electromagnetic fault injection: The curse of flip-flops. *Journal of Cryptographic Engineering* 7, 3 (Sep 2017), 183–197.

[162] Sıddıka Berna Örs, Elisabeth Oswald, and Bart Preneel. 2003. Power-analysis attacks on an FPGA—First experimental results. In *Cryptographic Hardware and Embedded Systems*. Springer, Cologne, Germany, 35–50.

[163] David Oswald and Christof Paar. 2012. Improving side-channel analysis with optimal linear transforms. In *International Conference on Smart Card Research and Advanced Applications*. Springer, Graz, Austria, 219–233.

[164] Colin O'Flynn and Alex Dewar. 2019. On-device power analysis across hardware security domains: Stop hitting yourself. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019, 4 (Aug 2019), 126–153.

[165] Jungmin Park, Fahim Rahman, Apostol Vassilev, Domenic Forte, and Mark Tehranipoor. 2019. Leveraging side-channel information for disassembly and security. *J. Emerg. Technol. Comput. Syst.* 16, 1, Article 6 (Dec. 2019), 21 pages.

[166] Jungmin Park, Xiaolin Xu, Yier Jin, Domenic Forte, and Mark Tehranipoor. 2018. Power-based side-channel instruction-level disassembler. In *55th ACM/ESDA/IEEE Design Automation Conference*. IEEE, San Francisco, CA, 1–6.

[167] Gilles Piret, Thomas Roche, and Claude Carlet. 2012. PICARO - A block cipher allowing efficient higher-order side-channel resistance. In *International Conference on Applied Cryptography and Network Security*. Springer, Singapore, 311–328.

[168] Robert Primas, Peter Pessl, and Stefan Mangard. 2017. Single-trace side-channel attacks on masked lattice-based encryption. In *Cryptographic Hardware and Embedded Systems*. Springer, Taipei, Taiwan, 513–533.

[169] George Provelengios, Daniel Holcomb, and Russell Tessier. 2019. Characterizing power distribution attacks in multi-user FPGA environments. In *29th International Conference on Field-Programmable Logic and Applications*. IEEE, Barcelona, Spain, 194–201.

[170] George Provelengios, Daniel Holcomb, and Russel Tessier. 2020. Power wasting circuits for cloud FPGA attacks. In *30th International Conference on Field-Programmable Logic and Applications*. IEEE, Gothenburg, Sweden, 231–235.

[171] Rui Qiao and Mark Seaborn. 2016. A new approach for RowHammer attacks. In *International Symposium on Hardware-Oriented Security and Trust*. IEEE, McLean, VA, 161–166.

[172] Yi Qin and Chuan Yue. 2018. Website fingerprinting by power estimation based side-channel attacks on Android 7. In *17th International Conference on Trust, Security And Privacy in Computing and Communications/12th International Conference on Big Data Science and Engineering (TrustCom/BigDataSE'18)*. IEEE, New York, NY, 1030–1039.

[173] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. 2019. VoltJockey: Breaching trustzone by software-controlled voltage manipulation over multi-core frequencies. In *Conference on Computer and Communications Security*. ACM, London, UK, 195–209.

[174] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. 2019. VoltJockey: Breaking SGX by software-controlled voltage-induced hardware faults. In *Asian Hardware Oriented Security and Trust Symposium (AsianHOST'19)*. IEEE, Xi'an, China, 1–6.

[175] Jean-Jacques Quisquater and David Samyde. 2002. Automatic code recognition for smart cards using a Kohonen neural network. In *International Conference on Smart Card Research and Advanced Applications*. USENIX, San Jose, CA, 9 pages.

[176] Masudul Hassan Quraishi, Erfan Bank Tavakoli, and Fengbo Ren. 2021. A Survey of System Architectures and Techniques for FPGA Virtualization. Retrieved December 2, 2021 from arXiv:2011.09073 [cs.AR]

[177] Chethan Ramesh, Shivukumar B. Patil, Siva Nishok Dhanuskodi, George Provelengios, Sebastien Pillement, Daniel Holcomb, and Russell Tessier. 2018. FPGA side channel attacks without physical access. In *26th Symposium on Field-Programmable Custom Computing Machines*. IEEE, Boulder, CO, 45–52.

[178] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. 2020. SCAUL: Power side-channel analysis with unsupervised learning. *IEEE Trans. Comput.* 69, 11 (Nov 2020), 1626–1638.

[179] Mark Randolph and William Diehl. 2020. Power side-channel attack analysis: A review of 20 years of study for the layman. *Cryptography* 4, 2 (May 2020), 33 pages.

[180] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. 2016. Flip feng shui: Hammering a needle in the software stack. In *25th Security Symposium*. USENIX, Austin, TX, 1–18.

[181] Francesco Regazzoni, Wang Yi, and François-Xavier Standaert. 2011. FPGA Implementations of the AES masked against power analysis attacks. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Darmstadt, Germany, 1–11.

[182] Lionel Rivière, Zakaria Najm, Pablo Rauzy, Jean-Luc Danger, Julien Bringer, and Laurent Sauvage. 2015. High precision fault injections on the instruction cache of ARMv7-M architectures. In *International Symposium on Hardware-Oriented Security and Trust*. IEEE, Washington, DC, 62–67.

[183] Pieter Robyns, Mariano Di Martino, Dennis Giese, Wim Lamotte, Peter Quax, and Guevara Noubir. 2020. Practical operation extraction from electromagnetic leakage for side-channel analysis and reverse engineering. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, Linz, Austria, 161–172.

[184] Ryan Roemer, Erik Buchanan, Hovav Shacham, and Stefan Savage. 2012. Return-oriented programming: Systems, languages, and applications. *ACM Trans. Inf. Syst. Secur.* 15, 1, Article 2 (March 2012), 34 pages.

[185] Majid Sabbagh, Yunsi Fei, and David Kaeli. 2020. A novel GPU overdrive fault attack. In *57th ACM/ESDA/IEEE Design Automation Conference*. IEEE, San Francisco, CA, 1–6.

[186] Behzad Salami, Erhan Baturay Onural, Ismail Emir Yuksel, Fahrettin Koc, Oguz Ergin, Adrián Cristal Kestelman, Osman Unsal, Hamid Sarbazi-Azad, and Onur Mutlu. 2020. An experimental study of reduced-voltage operation in modern FPGAs for neural network acceleration. In *50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, Valencia, Spain, 138–149.

[187] Pascal Sasdrich, Amir Moradi, Oliver Mischke, and Tim Güneysu. 2015. Achieving side-channel protection with dynamic logic reconfiguration on modern FPGAs. In *International Symposium on Hardware-Oriented Security and Trust*. IEEE, Washington, DC, 130–136.

[188] Asanka Sayakkara, Nhien-An Le-Khac, and Mark Scanlon. 2019. A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics. *Digital Investigation* 29 (Jun 2019), 43–54.

[189] Falk Schellenberg, Dennis R. E. Gnad, Amir Moradi, and Mehdi B. Tahoori. 2018. An inside job: Remote power analysis attacks on FPGAs. In *Design, Automation and Test in Europe*. IEEE, Dresden, Germany, 1111–1116.

[190] Falk Schellenberg, Dennis R. E. Gnad, Amir Moradi, and Mehdi B. Tahoori. 2018. Remote inter-chip power analysis side-channel attacks at board-level. In *IEEE/ACM International Conference on Computer-Aided Design*. IEEE, New York, NY, 114:1–114:7.

[191] Jörn-Marc Schmidt and Michael Hutter. 2007. Optical and EM fault-attacks on CRT-based RSA: Concrete results. In *15th Austrian Workshop on Microelectronics*. Verlag der Technischen Universität Graz, Graz, Austria, 61–67.

[192] Mark Seaborn and Thomas Dullien. 2015. Exploiting the DRAM RowHammer bug to gain kernel privileges. *Black Hat* 15 (March 2015), 71 pages.

[193] Nader Sehatbakhsh, Baki Berkay Yilmaz, Alenka Zajic, and Milos Prvulovic. 2020. A new side-channel vulnerability on modern computers by exploiting electromagnetic emanations from the power management unit. In *International Symposium on High Performance Computer Architecture (HPCA'20)*. IEEE, San Diego, CA, 123–138.

[194] Zeinab Seifoori, Seyedeh Sharareh Mirzargar, and Mirjana Stojilović. 2020. Closing leaks: Routing against crosstalk side-channel attacks. In *28th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM, Seaside, CA, 197–203.

[195] Bodo Selmke, Florian Hauschild, and Johannes Obermaier. 2019. Peak clock: Fault injection into PLL-based systems via clock manipulation. In *3rd Workshop on Attacks and Solutions in Hardware Security Workshop*. ACM, London, UK, 85–94.

[196] Seyed Mohammad Seyedzadeh, Alex K. Jones, and Rami Melhem. 2017. Counter-based tree structure for row hammering mitigation in DRAM. *IEEE Computer Architecture Letters* 16, 1 (Jan 2017), 18–21.

[197] Linda L. Shen, Ibrahim Ahmed, and Vaughn Betz. 2019. Fast voltage transients on FPGAs: Impact and mitigation strategies. In *27th Symposium on Field-Programmable Custom Computing Machines*. IEEE, San Diego, CA, 271–279.

[198] Mungyu Son, Hyunsun Park, Junwhan Ahn, and Sungjoo Yoo. 2017. Making DRAM stronger against row hammering. In *54th Annual Design Automation Conference*. ACM, Austin, TX, 1–6.

[199] Albert Spruyt, Alyssa Milburn, and Łukasz Chmielewski. 2021. Fault injection as an oscilloscope: Fault correlation analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021, 1 (2021), 192–216.

[200] François-Xavier Standaert, Sıddıka Berna Örs, Jean-Jacques Quisquater, and Bart Preneel. 2004. Power analysis attacks against FPGA implementations of the DES. In *14th International Conference on Field-Programmable Logic and Applications*. Springer, Leuven, Belgium, 84–94.

[201] John E. Stone, David Gohara, and Guochun Shi. 2010. OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in Science Engineering* 12, 3 (May 2010), 66–73.

[202] Edward Stott, Joshua M. Levine, Peter Y. K. Cheung, and Nachiket Kapre. 2014. Timing fault detection in FPGA-based circuits. In *22nd Symposium on Field-Programmable Custom Computing Machines*. IEEE, Boston, MA, 96–99.

[203] Stratix10. 2020. Intel® Stratix® 10 Power Management User Guide. Retrieved February 19, 2021 from intel.com.

[204] Daehyun Strobel, Florian Bache, David Oswald, Falk Schellenberg, and Christoph Paar. 2015. SCANDALee: A side-ChANnel-based DisAssembLer using Local Electromagnetic Emanations. In *Design, Automation and Test in Europe*. IEEE, Grenoble, France, 139–144.

[205] Go Takatoi, Takeshi Sugawara, Kazuo Sakiyama, and Yang Li. 2020. Simple electromagnetic analysis against activation functions of deep neural networks. In *International Conference on Applied Cryptography and Network Security*. Springer, Rome, Italy, 181–197.

[206] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo. 2017. CLKSCREW: Exposing the perils of security-oblivious energy management. In *26th Security Symposium*. USENIX, Vancouver, BC, 1057–1074.

[207] Andrei Tatar, Radhesh Krishnan Konoth, Elias Athanasopoulos, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Throwhammer: Rowhammer attacks over the network and defenses. In *Annual Technical Conference*. USENIX, Boston, MA, 213–26.

[208] Tempest. 1972. TEMPEST: A Signal Problem. Retrieved December 2, 2021 from https://cryptome.org/nsa-tempest.pdf.

[209] Niek Timmers and Cristofaro Mune. 2017. Escalating privileges in Linux using voltage fault injection. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Taipei, Taiwan, 1–8.

[210] Thomas Trouchkine, Guillaume Bouffard, and Jessy Clédière. 2019. Fault injection characterization on modern CPUs. In *IFIP International Conference on Information Security Theory and Practice*. Springer, Paris, France, 123–138.

[211] Furkan Turan and Ingrid Verbauwhede. 2020. Trust in FPGA-accelerated cloud computing. *ACM Comput. Surv.* 53, 6, Article 128 (Dec. 2020), 28 pages.

[212] Shahram Vafa, Massoud Masoumi, and Amir Amini. 2020. An efficient profiling attack to real codes of PIC16F690 and ARM Cortex-M3. *IEEE Access* 8 (Dec 2020), 222520–222532.

[213] V. M. Vaidyan and Akhilesh Tyagi. 2020. Instruction level disassembly through electromagnetic side-channel: Machine learning classification approach with reduced combinatorial complexity. In *3rd International Conference on Signal Processing and Machine Learning*. ACM, Beijing, China, 124–130.

[214] Anuj Vaishnav, Khoa Dang Pham, and Dirk Koch. 2018. A survey on FPGA virtualization. In *28th International Conference on Field-Programmable Logic and Applications*. IEEE, Dublin, Ireland, 131–138.

[215] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clementine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. 2016. Drammer: Deterministic RowHammer attacks on mobile platforms. In *Conference on Computer and Communications Security*. ACM, Vienna, Austria, 1675–1689.

[216] Victor van der Veen, Martina Lindorfer, Yanick Fratantonio, Harikrishnan Padmanabha Pillai, Giovanni Vigna, Christopher Kruegel, Herbert Bos, and Kaveh Razavi. 2018. GuardION: Practical mitigation of DMA-based RowHammer attacks on ARM. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, Saclay, France, 92–113.

[217] vGPU 2021. Virtual GPU Software User Guide. Retrieved August 24, 2021 from nvidia.com.

[218] Saru Vig, Sarani Bhattacharya, Debdeep Mukhopadhyay, and Siew-Kei Lam. 2018. Rapid detection of RowHammer attacks using dynamic skewed hash tree. In *7th International Workshop on Hardware and Architectural Support for Security and Privacy*. ACM, Los Angeles, CA, 1–8.

[219] Chao Wang and Patrick Schaumont. 2017. Security by compilation: An automated approach to comprehensive side-channel resistance. *ACM SIGLOG News* 4, 2 (May 2017), 76–89.

[220] Huanyu Wang and Elena Dubrova. 2020. Tandem Deep Learning Side-Channel Attack Against FPGA Implementation of AES. Cryptology ePrint Archive, Report 2020/373. https://eprint.iacr.org/2020/373.

[221] Gary Wassermann and Zhendong Su. 2007. Sound and precise analysis of web applications for injection vulnerabilities. *SIGPLAN Not.* 42, 6 (June 2007), 32–41.

[222] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. 2018. I know what you see: Power side-channel attack on convolutional neural network accelerators. In *34th Annual Computer Security Applications Conference*. ACM, San Juan, PR, 393–406.

[223] Léo Weissbart, Stjepan Picek, and Lejla Batina. 2019. One trace is all it takes: Machine learning-based side-channel attack on edDSA. In *Security, Privacy, and Applied Cryptography Engineering*. Springer, Gandhinagar, India, 86–105.

[224] Zane Weissman, Thore Tiemann, Daniel Moghimi, Evan Custodio, Thomas Eisenbarth, and Berk Sunar. 2020. JackHammer: Efficient RowHammer on heterogeneous FPGA-CPU platforms. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 3 (Jun 2020), 169–195.

[225] Lin Yan, Yao Guo, Xiangqun Chen, and Hong Mei. 2015. A study on power side channels on mobile devices. In *7th Asia-Pacific Symposium on Internetware*. ACM, Wuhan, China, 30–38.

[226] Yuan Yao, Mo Yang, Conor Patrick, Bilgiday Yuce, and Patrick Schaumont. 2018. Fault-assisted side-channel analysis of masked implementations. In *International Symposium on Hardware-Oriented Security and Trust*. IEEE, Washington, DC, 57–64.

[227] Abdullah Giray Yağlıkčı, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu. 2021. BlockHammer: Preventing RowHammer at low cost by blacklisting rapidly-accessed DRAM rows. In *International Symposium on High-Performance Computer Architecture (HPCA'21)*. IEEE, Seoul, Korea, 345–358.

[228] Sadegh Yazdanshenas and Vaughn Betz. 2019. The costs of confidentiality in virtualized FPGAs. *IEEE Trans. on VLSI* 27 (Oct. 2019), 2272–2283.

[229] Baki Berkay Yilmaz, Robert L. Callan, Milos Prvulovic, and Alenka Zajić. 2018. Capacity of the EM covert/side-channel created by the execution of instructions in a processor. *IEEE Trans. Inf. Forensics Secur.* 13, 3 (Mar 2018), 605–620.

[230] Kota Yoshida, Takaya Kubota, Mitsuru Shiozaki, and Takeshi Fujino. 2019. Model-extraction attack against FPGA-DNN accelerator utilizing correlation electromagnetic analysis. In *27th Symposium on Field-Programmable Custom Computing Machines*. IEEE, San Diego, CA, 318.

[231] Bilgiday Yuce, Nahid Farhady Ghalaty, Harika Santapuri, Chinmay Deshpande, Conor Patrick, and Patrick Schaumont. 2016. Software fault resistance is futile: Effective single-glitch attacks. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, Santa Barbara, CA, 47–58.

[232] Bilgiday Yuce, Patrick Schaumont, and Marc Witteman. 2018. Fault attacks on secure embedded software: Threats, design, and evaluation. *Journal of Hardware and Systems Security* 2, 2 (Jun 2018), 111–130.

[233] Shaza Zeitouni, Ghada Dessouky, and Ahmad-Reza Sadeghi. 2020. SoK: On the Security Challenges and Risks of Multi-Tenant FPGAs in the Cloud. Retrieved December 2, 2021 from https://arxiv.org/abs/2009.13914.

[234] Yicheng Zhang, Rozhin Yasaei, Hao Chen, Zhou Li, and Mohammad Abdullah Al Faruque. 2021. Stealing neural network structure through remote FPGA side-channel analysis. In *29th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM, Virtual, 225.

[235] Zhenkai Zhang, Zihao Zhan, Daniel Balasubramanian, Bo Li, Peter Volgyesi, and Xenofon Koutsoukos. 2020. Leveraging EM side-channel information to detect RowHammer attacks. In *IEEE Symposium on Security and Privacy (S&P'20)*. IEEE, San Francisco, CA, 729–746.

[236] Mark Zhao and G. Edward Suh. 2018. FPGA-based remote power side-channel attacks. In *IEEE Symposium on Security and Privacy (S&P'18)*. IEEE, San Francisco, CA, 805–820.

[237] YongBin Zhou and DengGuo Feng. 2005. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *IACR Cryptol. ePrint Arch.* 2005 (Jan. 2005), 34 pages.

[238] Kenneth M. Zick and John P. Hayes. 2012. Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems. *ACM Trans. on Reconfigurable Technology and Systems* 5, 1 (Mar 2012), 1:1–1:26.

[239] Kenneth M. Zick, Meeta Srivastav, Wei Zhang, and Matthew French. 2013. Sensing nanosecond-scale voltage attacks and natural transients in FPGAs. In *21st ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM, Monterey, CA, 101–104.

[240] Loïc Zussa, Amine Dehbaoui, Karim Tobich, Jean-Max Dutertre, Philippe Maurine, Ludovic Guillaume-Sage, Jessy Clediève, and Assia Tria. 2014. Efficiency of a glitch detector against electromagnetic fault injection. In *Design, Automation and Test in Europe*. IEEE, Dresden, Germany, 1–6.

[241] Loïc Zussa, Jean-Max Dutertre, Jessy Clédière, and Bruno Robisson. 2014. Analysis of the fault injection mechanism related to negative and positive power supply glitches using an on-chip voltmeter. In *International Symposium on Hardware-Oriented Security and Trust*. IEEE, Arlington, VA, 130–135.

[242] 2020. Zynq UltraScale+ Device Technical Reference Manual. Retrieved February 8, 2021 from xilinx.com.

Received April 2021; revised August 2021; accepted November 2021