Remote Attacks on FPGA Hardware

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Dennis Rolf Engelbert Gnad

aus Pforzheim, Deutschland

Tag der mündlichen Prüfung:	15.05.2020
1. Referent:	Prof. Dr. Mehdi B. Tahoori, Karlsruher Institut für Technologie (KIT)
2. Referent:	Priv.-Doz. Dr. Amir Moradi, Ruhr-Universität Bochum (RUB)

Acknowledgments

Herewith I thank all persons that have supported me towards pursuing and finishing my PhD. Without many of you, this thesis would not have been possible. You were there as friends, partner, family, supervisors, or any other role which helped me to either concentrate on the thesis itself, or provide me with the right environment.

First of all I would like to thank my supervisor Prof. Dr. Mehdi B. Tahoori. Without him, this thesis would not even have started, and certainly not finished. He always supported me in all of the steps, from paper writing to brainstorming of new experiments, and I learned a lot from him. The same goes for Dr.-Ing. Fabian Oboril, who advised me in the first years of the thesis, where Dr.-Ing. Saman Kiamehr was also very helpful. A very important role was also played by the second advisor of my thesis, Priv.-Doz Dr. Amir Moradi from Ruhr-Universität Bochum (RUB). Together with Dr.-Ing. Falk Schellenberg (also from RUB), they introduced me to the topics of side-channel attacks. With both of them, I have co-authored several papers. Furthermore, I would like to thank Jonas Krautter, who joined the Chair of Dependable Nanocomputing (CDNC) as a PhD Student after his Master Thesis with us. Together with him, we also published several papers. In one publication we also got support by Kevin Schäfer from Rutronik Elektronische Bauelemente GmbH. He sponsored us with a few boards for one of our works, and also helped us with his expertise.

Furthermore, I would like to thank all my other colleagues as well. It was always nice to have your company when going to lunch. The weekly meetings were also very interesting to get insights into the work of others. Furthermore, they helped in our well-organized internal reviews, such that papers could be improved before peer review. I would like to especially thank our secretary Iris Schröder-Piepka, who is always very kind and helpful regarding various organizational things, for instance looking for our students doing a thesis or an assistant job, or help in planning business trips.

I should also mention that before I even started the PhD, I worked at the same Institute of Computer Engineering (ITEC) on my Master Thesis, but at the Chair of Embedded Systems (CES) of Prof. Dr.-Ing. Jörg Henkel under the supervision of Prof. Dr.-Ing. Muhammad Shafique (at that time Postdoc). If they had not introduced me to academic research and publishing, I would probably not have started my PhD.

Last but not least, I thank all my friends and family, who have always been there for me. Without the support and encouragement of my parents and grandparents I would have never started this endeavor. Without my brother and various friends around me, there would not have been a good balance between life and work. Especially during the last months of writing the thesis and defending the PhD, there were additional difficulties for me, and the global outbreak of COVID-19 did not make them easier. Without writing all their names, I thank anyone who was there during these last months, even if it was just by providing good company. :-)

Dennis Gnad
Höhenstraße 10
75210 Keltern

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben haben und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen – die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Karlsruhe, 15. Mai 2020
Dennis R. E. Gnad

Abstract

An increasing amount of computer systems are connected on a global scale and become remotely accessible, increasing their requirements for security. One recent technology that is increasingly used as a computing accelerator, both for embedded systems and in the cloud, are *Field-Programmable Gate Arrays* (FPGAs). They are very flexible devices that can be configured and programmed by software, to implement arbitrary digital circuits. Like other integrated circuits, FPGAs are based on modern semiconductor technologies that are affected by variations in the manufacturing process and runtime conditions. It is already known that these variations impact the reliability of a system, but their impact on security has not been widely explored.

This PhD thesis looks into a cross section of these topics: Remotely-accessible and multi-user FPGAs, and the security threats dependent on physical variation in modern semiconductor technologies. The first contribution in this thesis identifies transient voltage fluctuations as one of the highest impacts on FPGA performance, and experimentally analyzes their dependency on the workload the FPGA is executing. In the remaining thesis, the security implications of these transient voltage fluctuations are explored. Various attacks are proven possible that were previously thought to require physical access to the chip, and the use of dedicated and expensive test and measurement equipment. It shows that isolation countermeasures can be circumvented by a malicious user with partial access to the FPGA, affecting other users in the same FPGA, or complete system.

Using circuits to affect the FPGA on-chip voltage, active attacks are shown that can cause faults in other parts of the system. By that, Denial-of-Service is possible, and can also be escalated to extract secret key information from the system. Furthermore, passive attacks are shown that indirectly measure the on-chip voltage fluctuations, and show that these measurements are sufficient to extract secret key information through power analysis side-channel attacks, which can also be escalated to other chips connected to the same power supply as the FPGA. To prove comparable attacks are not exclusive to FPGAs, small IoT devices are also shown vulnerable to attacks that leverage on partial access to their power distribution network.

Overall, this thesis shows that fundamental physical variations in integrated circuits can undermine the security of an entire system, even if the attacker is absent from the device. For FPGAs in their current form, these problems need to be solved before they can be securely used in multi-user systems, or with 3rd party access to them. In publications that are not part of this thesis, some first countermeasures were already explored.

Zusammenfassung

Immer mehr Computersysteme sind weltweit miteinander verbunden und über das Internet zugänglich, was auch die Sicherheitsanforderungen an diese erhöht. Eine neuere Technologie, die zunehmend als Rechenbeschleuniger sowohl für eingebettete Systeme als auch in der Cloud verwendet wird, sind *Field-Programmable Gate Arrays* (FPGAs). Sie sind sehr flexible Mikrochips, die per Software konfiguriert und programmiert werden können, um beliebige digitale Schaltungen zu implementieren. Wie auch andere integrierte Schaltkreise basieren FPGAs auf modernen Halbleitertechnologien, die von Fertigungstoleranzen und verschiedenen Laufzeitschwankungen betroffen sind. Es ist bereits bekannt, dass diese Variationen die Zuverlässigkeit eines Systems beeinflussen, aber ihre Auswirkungen auf die Sicherheit wurden nicht umfassend untersucht.

Diese Doktorarbeit befasst sich mit einem Querschnitt dieser Themen: Sicherheitsprobleme die dadurch entstehen wenn FPGAs von mehreren Benutzern benutzt werden, oder über das Internet zugänglich sind, in Kombination mit physikalischen Schwankungen in modernen Halbleitertechnologien. Der erste Beitrag in dieser Arbeit identifiziert transiente Spannungsschwankungen als eine der stärksten Auswirkungen auf die FPGA-Leistung und analysiert experimentell wie sich verschiedene Arbeitslasten des FPGAs darauf auswirken. In der restlichen Arbeit werden dann die Auswirkungen dieser Spannungsschwankungen auf die Sicherheit untersucht. Die Arbeit zeigt, dass verschiedene Angriffe möglich sind, von denen früher angenommen wurde, dass sie physischen Zugriff auf den Chip und die Verwendung spezieller und teurer Test- und Messgeräte erfordern. Dies zeigt, dass bekannte Isolationsmaßnahmen innerhalb FPGAs von böswilligen Benutzern umgangen werden können, um andere Benutzer im selben FPGA oder sogar das gesamte System anzugreifen.

Unter Verwendung von Schaltkreisen zur Beeinflussung der Spannung innerhalb eines FPGAs zeigt diese Arbeit aktive Angriffe, die Fehler (Faults) in anderen Teilen des Systems verursachen können. Auf diese Weise sind *Denial-of-Service* Angriffe möglich, als auch *Fault*-Angriffe um geheime Schlüsselinformationen aus dem System zu extrahieren. Darüber hinaus werden passive Angriffe gezeigt, die indirekt die Spannungsschwankungen auf dem Chip messen. Diese Messungen reichen aus, um geheime Schlüsselinformationen durch *Power Analysis* Seitenkanalangriffe zu extrahieren. In einer weiteren Eskalationsstufe können sich diese Angriffe auch auf andere Chips auswirken die an dasselbe Netzteil angeschlossen sind wie der FPGA. Um zu beweisen, dass vergleichbare Angriffe nicht nur innerhalb FPGAs möglich sind, wird gezeigt, dass auch kleine IoT-Geräte anfällig für Angriffe sind welche die gemeinsame Spannungsversorgung innerhalb eines Chips ausnutzen.

Insgesamt zeigt diese Arbeit, dass grundlegende physikalische Variationen in integrierten Schaltkreisen die Sicherheit eines gesamten Systems untergraben können, selbst wenn der Angreifer keinen direkten Zugriff auf das Gerät hat. Für FPGAs in ihrer aktuellen Form müssen diese Probleme zuerst gelöst werden, bevor man sie mit mehreren Benutzern oder mit Zugriff von Drittanbietern sicher verwenden kann. In Veröffentlichungen die nicht Teil dieser Arbeit sind wurden bereits einige erste Gegenmaßnahmen untersucht.

Contents

List of own publications included in this thesis

Transactions & Articles

[19] D. R. E. Gnad, F. Oboril, S. Kiamehr, M. B. Tahoori, "An Experimental Evaluation and Analysis of Transient Voltage Fluctuations in FPGAs", in IEEE Transactions on Very Large Scale Integration Systems (TVLSI), 2018.

[21] J. Krautter, D. R. E. Gnad, M. B. Tahoori, "FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, suitable for DFA on AES", in IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES), 2018. (CSAW'18 Finalist)

[24] D. R. E. Gnad, J. Krautter, M. B. Tahoori, "Leaky Noise: New Side-Channel Attack Vectors in Mixed-Signal IoT Devices", in IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES), 2019.

[25] D. R. E. Gnad, F. Schellenberg, J. Krautter, A. Moradi and M. B. Tahoori, "Remote Electrical-level Security Threats to Multi-Tenant FPGAs," in IEEE Design & Test, 2020.

Conferences

[18] D. R. E. Gnad, F. Oboril, S. Kiamehr, M. B. Tahoori, "Analysis of Transient Voltage Fluctuations in FPGAs", International Conference on Field-Programmable Technology (FPT), 2016, China. (Best Paper Candidate)

[20] D. R. E. Gnad, F. Oboril, M. B. Tahoori, "Voltage Drop-based Fault Attacks on FPGAs using Valid Bitstreams", International Conference on Field-Programmable Logic and Applications (FPL), 2017, Belgium. (Best Paper Award, *Stamatis Vassiliadis Award)

[22] F. Schellenberg, D. R. E. Gnad, A. Moradi, M. B. Tahoori, "An Inside Job: Remote Power Analysis Attacks on FPGAs", in proceedings of Design, Automation & Test in Europe (DATE), 2018, Germany. (Best Paper Candidate)

[23] F. Schellenberg, D. R. E. Gnad, A. Moradi, M. B. Tahoori, "Remote Inter-Chip Power Analysis Side-Channel Attacks at Board-Level", In Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2018, USA.

List of other publications not included in this thesis

[188] D. R. E. Gnad, S. Rapp, J. Krautter, M. B. Tahoori, "Checking for Electrical Level Security Threats in Bitstreams for Multi-Tenant FPGAs", International Conference on Field-Programmable Technology (FPT), 2018, Japan.

[187] J. Krautter, D. R. E. Gnad, F. Schellenberg, A. Moradi, and M. B. Tahoori, "Active Fences against Voltage-based Side Channels in Multi-Tenant FPGAs", in Proceedings of the International Conference on Computer-Aided Design (ICCAD), 2019, USA.

[189] J. Krautter, D. R. E. Gnad, M. B. Tahoori, "Mitigating Electrical-Level Attacks towards Secure Multi-Tenant FPGAs in the Cloud", in ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2019.

Part I.

Preliminaries

1. Introduction

Integrated circuits (ICs) based on semiconductors are one of the driving technologies of the 21st century. Next to computing accelerators based on Graphics Processing Units (GPUs), performance and energy efficiency can be further increased when custom computing accelerators in the form of Field Programmable Gate Array (FPGA) are added. Through reconfiguration, FPGAs are very flexible devices that can be configured and programmed by software to implement arbitrary digital circuits, and thus also accelerate most algorithms. FPGAs are becoming widespread in various embedded systems, personal computers, and high-end servers [1–5]. Microsoft is already using FPGAs in their datacenters since 2014 [6], and companies such as Amazon [7], Alibaba [8], and Huawei [9] are renting FPGAs as computing accelerators to arbitrary customers. Furthermore, we are heading towards a future where any device gets connected to the *Internet of Things*, either as small embedded devices or in datacenters for *Cloud Computing*, also putting them at higher security risk [10]. Both, the trend of cloud computing, as well as Internet of Things devices, are applied more and more in critical application domains. For instance, these consist of applications in healthcare, transportation, or public infrastructure, just to name a few.

Typically, most systems have been attacked by flaws in the software implementation. However, we also see an increase in attacks that leverage on properties or flaws in the hardware, undermining any security that software can provide. These problems are gaining more widespread attention, especially since the microarchitectural attacks named *Spectre* and *Meltdown* were revealed in January 2018 [11, 12]. Thus, hardware technologies, such as FPGAs, need to be more thoroughly analyzed in their security aspects, before they get widely adopted as computing accelerators. A single FPGA can be virtualized and shared among multiple users and tasks, opening new questions how isolated these users are from each other. Potential security risks on the hardware level of virtualized FPGAs have been mostly disregarded prior to this thesis.

In this PhD thesis, we thus focus on new types of security threats when FPGAs are used as computing accelerators, or parts of any bigger system. Modern semiconductor technologies suffer from higher physical variations in the manufacturing process and changing runtime conditions. Such variations are typically handled with increased safety margins, sufficient for typical operating conditions [13, 14]. One of the highest and fastest variations are changes in the power supply voltage level within the chip, which can change as rapidly as single clock cycles of the operating clock of the system [15–17]. It is already well-known that these transient voltage fluctuations impact the reliability of a system, but their effect on security has not been widely explored prior to this thesis. Thus, we experimentally analyze on-chip voltage fluctuations and then show that they can indeed lead to new security threats for FPGA-based systems. To the best of our knowledge, these new threats are also the first to show that physical attacks do not necessarily need local access to the device. Through various indirect ways, voltage fluctuations can be caused or measured, making it feasible for power analysis or fault attacks to be performed remotely. While the main focus is on FPGAs, the thesis also proves voltage fluctuations to be an issue in mixed-signal integrated circuits, or inside a printed circuit board.

1.1. Contributions

The main contribution of this PhD thesis is showing that voltage-based side channel and fault attacks are feasible only through software access to a system. By that, it also becomes feasible remotely.

During the PhD thesis, methods were successfully developed to remotely extract side-channel leakage or cause faults on the electrical level of various FPGA chips, and some mixed-signal devices used in low-cost IoT applications. Attacks that previously required dedicated test and measurement equipment can now be performed in software, potentially remotely. By that, to the best of our knowledge, the thesis proves for the first time that power analysis side-channel attacks or fault attacks can also be performed remotely, and need to be considered in threat models that previously ignored them.

More specifically, this thesis shows power analysis attacks to be a potential security risk in multi-user systems with FPGAs and highly-integrated mixed-signal IoT systems. For instance, such systems are FPGAs virtualized as multi-tenant systems, or IoT applications that do not limit access to analog sensors in the system.

The following subsections explains the individual contributions and clarifies what has been done in respect to the co-authors of the respective works that have been published already.

1.1.1. On-Chip Characterization of Voltage Fluctuations

The thesis experimentally confirms that voltage fluctuations have one of the highest impacts on circuit delay, being a potential risk for both system stability as well as side-channel leakage. These nanosecond-scale changes can not be simulated easily on chip level, and thus, the flexibility of FPGAs is used to perform indirect measurements. Subsequently, further details on temporal and spatial changes in on-chip voltage was analyzed, and how it is impacted from various workload activities on the FPGA, similar to real-time workloads. This led to the discovery of previously unreported workload-dependent influences, which can now be used for FPGA design optimization. Some of these insights were only found because the full system and all of its characteristics were analyzed at once, instead of just individual components of the system. Based on these methods, the security implications of voltage fluctuations were analyzed in the following.

This contribution was published in [18], and has been extended in [19]. The most part of the respective Chapter 3 is identical with [19]. All experiments have been performed by the author of this thesis and have been planned with advise from co-authors of that publication: Fabian Oboril, Saman Kiamehr, and Mehdi Tahoori.

1.1.2. Adversary Model for Multi-Tenant FPGAs

The thesis first introduced that power analysis side-channel attacks or fault attacks can be a threat for remotely accessible FPGAs, i.e. introduced only by software-defined FPGA configuration. The first general idea of this threat or adversary model was published in [20] (Section 4.1). It has been gradually adjusted in the subsequent publications, which are part of this thesis [21–24] (Chapter 4, Chapter 5). Previously, such attacks were carried out through physical access and with dedicated fault injection and measurement equipment. In multi-tenant FPGA operation, multiple users share the same FPGA, geometrically separated into individual regions. In that scenario, the named attacks are serious security threats that need to be solved before widespread multi-tenant FPGA operation is feasible.

The detailed threat or adversary models are explained in the respective sections of Chapter 4 and Chapter 5.

1.1.3. PDN-based Voltage Drop-based Fault Attacks

In a followup contribution, a malicious user in one part of an FPGA is shown to either perform a Denial-of-Service (DoS) attack on the full system, or cause precise faults in the design of another isolated user on the same FPGA. All of that is shown possible through software-defined configuration, by crafting logic blocks that consume high current, and then toggle them in respective patterns. Next to DoS, the fault precision is enough to affect individual rounds of an Advanced Encryption Standard (AES) module, allowing Differential Fault Analysis (DFA) to extract secret keys.

These contributions have been published in [20], [21], and [25], on which Section 4.1, Section 4.2, and Section 4.3 are based on, respectively. The experiments for [21] (Section 4.2) have been jointly devised and performed with Jonas Krautter, while the experiments for Section 4.3 have partially been devised and performed together with Falk Schellenberg of the Ruhr-Universität Bochum.

1.1.4. PDN-based Power Side-Channel Analysis Attacks

In a collaboration with researchers Falk Schellenberg and Amir Moradi of the Ruhr-Universität Bochum this thesis shows that the sensors based on FPGA fabric can sufficiently measure on-chip voltage fluctuations to perform power analysis side-channel attacks inside the chip. When multiple voltage traces are recorded during the operation of an AES module, the secret key can be extracted using Correlation Power Analysis (CPA). That means, in a similar scenario as for fault attacks, an attacker residing in one part of the FPGA can extract the secrets from a victim in another part of the FPGA. Furthermore, the thesis shows how such attacks can be escalated beyond FPGAs, to other chips on the same Printed Circuit Board (PCB). It shows how secret keys from the AES and Rivest-Shamir-Adleman (RSA) cipher of that other chip can be extracted. This is again achieved with the software-defined on-chip sensors and does not require logical connections to the victim, proving an additional threat from insufficient supply chain security or insecure firmware updates. I.e. firmware updates alone can introduce a power side-channel attacker who previously had to connect dedicated measurement equipment to the device under attack. These contributions have been published in [22] and [23], which have been integrated into this thesis as Section 5.1 and Section 5.2. The experiments for these works have been jointly performed with Falk Schellenberg of the Ruhr-Universität Bochum.

To prove the generality of on-chip voltage fluctuations as a security vulnerability, other computing devices were also analyzed. Low cost devices used in Internet-of-Things applications often integrate analog and digital components on a single chip. It is shown that data recorded with the analog subsystem of the chip can contain secret information from the digital subsystem in form of correlated noise, which is proven with a Correlation Power Analysis (CPA) attack on AES. That contribution has been published in [24], on which Section 5.3 is based on. The experiments for this work have been jointly devised and performed with Jonas Krautter.

1.1.5. Main High-Level Contribution

The main contribution of this thesis is generating an awareness in the research community that physical attacks do not necessarily need local access to the device under attack. This has been shown with fault and power analysis side-channel attacks within FPGAs, but also another chip, showing that this can be a general problem for all modern semiconductor devices, and motivates further research in this direction.

1.2. Outline

The remaining thesis is organized in the following chapters:

- Chapter 2 summarizes basic background knowledge on the implementation of power supply networks for semiconductor chips, on-chip voltage fluctuations, common application scenarios, power analysis side-channel attacks, and fault attacks.
- Chapter 3 analyzes temporal and spatial on-chip voltage fluctuations in FPGAs.
- Chapter 4 shows how faults can be maliciously introduced through software-only configuration into FPGAs, to perform DoS and DFA.
- Chapter 5 shows how power analysis side-channel attacks can be performed through software access inside an FPGA chip, in other chips with a shared power supply, and even across the full board-level.
- Chapter 6 discusses related work with a relation to this thesis.
- Chapter 7 concludes the thesis and gives a perspective on further research directions.

2. Background

The sections in this chapter were partially overtaken from previously published works included in this thesis, which were co-authored with (in no particular order): Falk Schellenberg, Jonas Krautter, Fabian Oboril, Saman Kiamehr, Amir Moradi and Mehdi B. Tahoori.

This chapter explains and presents relevant background knowledge that is required to understand the remaining chapters of the thesis, mainly in three areas:

- The underlying electrical implementation of semiconductor chips
- Applications and related threat models in which FPGAs and Microcontrollers are used
- Analysis of side-channel data that can be gathered from these lower implementation layers

The contents of this chapter have been taken from the respective publications included inside this thesis [19–23, 25], with minor adjustments to fit in this format.

2.1. Power Distribution Networks

Each modern electronic system requires at least one power supply, integrated on the PCB. In total, a Power Distribution Network (PDN) starts at a voltage regulator as the main board-level power supply at a higher voltage, e.g. 12V. The power is hierarchically distributed across the board and can go through multiple stages of lower voltage regulators. These regulators are often switched-mode *Voltage Regulator Modules* (VRMs) and supply a range of chips on the board. Finally, a complete system-level PDN also consists of several passive resistive, capacitive and inductive (RCL) components. Based on these RCL-networks, several works discuss two main types of supply voltage drop. One of them is mainly due to a static current bias by parasitic resistance (IR drop), the other depends on the change of current over time, biased by primarily inductive components (Ldi/dt drop) [16, 17, 26–28]. Resistance causes IR drop and can therefore both be observed with a steady load on average, or transiently.

Ldi/dt drop is transient in nature and thus requires a high enough sample rate to be perfectly observable, and can be as fast as the circuit operating frequency, or as low as the operating frequency of the switched-mode VRM. With advanced technology nodes, Ldi/dt-dependent transient voltage fluctuations are becoming one of the major concerns to chip manufacturers [16, 17, 27, 29]. A resonance between on-chip decoupling capacitors and resistive and inductive components can lead to even less voltage stability [15, 16]. Compared with other variations that can affect semiconductor circuit performance, such as manufacturing process or temperature variations, voltage fluctuations have one of the highest influences on the required timing margin, and change faster than other fluctuations (up to circuit speed).

The differences in the electrical current i required to cause a voltage drop is influenced by both spatial and temporal circuit switching activity, which in turn is dependent on workload characteristics and system behavior [15, 16, 30]. Subsequently, a maliciously crafted circuit and respective switching activity could lead to corner cases with insufficient voltage stability and jeopardize security. Rising system complexity can lead to more corner cases, which are left from time-consuming and complex electrical-level design validation, and elevate these risks.

If an overall path is affected by a voltage drop for a long enough time (min. threshold time T_t) and high enough amplitude (V_{drop}), timing faults occur. Such a voltage drop is also known as a *voltage emergency* [31, 32]. In addition to timing faults, SRAM bit cells can lose data when their static noise margin voltage is violated [33].

2.2. Switched-Mode Voltage Regulator Modules (VRM)

In the past, small electronic devices were predominately supplied from linear regulators. These regulators use a transistor to operate as a variable resistor, and thus are able to quickly adapt to load changes, but also dissipate the power difference as heat. Because of this inefficiency, modern digital circuits use a switched-mode VRM [34], which uses the transistor that conducts the power only in the ON or OFF state.

The disadvantage of switched-mode VRMs is that they operate in a discrete time domain, with operating frequencies usually below 2 MHz, limited by non-ideal components on the board, as well as parasitic components introduced through board layout considerations. Many digital circuits run in a MHz–GHz range, beyond the Nyquist-frequency of the switched-mode VRM. Thus, stabilizing beyond these frequencies can only be achieved by assuming a maximum expected change in load current. The speed limitations of switched-mode VRMs is addressed by integrating them into chip packages, which is still an ongoing engineering challenge. Intel's *fully-integrated voltage regulator* [35], is one of a few commercial devices that already apply them. However, they have not been integrated in FPGAs yet.

In the following Fig. 2.1, a basic step-down (buck) configuration of a switched-mode VRM from [34] is shown. The transistor *Q1* is operated as the switching transistor that is completely closed or open, where only minimum energy is burned through its $R_{DS,on}$ resistance [34, 36]. In each discrete sampling step, the output voltage V_O is sampled and compared against the demanded voltage, deciding the next duty cycle, how long *Q1* is enabled. When *Q1* is enabled, the inductor current I_L is charged. When the switch of *Q1* is open, the inductor current goes through diode $D1$, while the voltage V_O is maintained through the current stored in inductor L. As described in [34], the respective duty cycle of the on-time of the transistor T_{ON} over the period of one sampling period $T_S = (T_{ON} + T_{OFF})$ leads to an *average* DC output voltage $V_{O(DC)}$ of:

$$V_{O(DC)} = AVG[V_{SW}] = \frac{T_{ON}}{T_S} \cdot V_{IN}$$

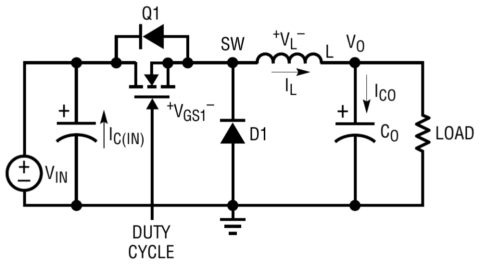


Figure 2.1.: Basic operating mode of a step-down (buck) switched-mode VRM, taken from [34].

To stabilize V_O between samples, inductor L and capacitor C_O are added at the output of the voltage regulator, which are sized for the amount of expected discharge between samples ΔQ. That translates to a relationship with the specific maximum peak current $I_{pk} = max(I_L)$. Depending on the capacitor size C_O, this results in a current-related peak-to-peak voltage ripple V_{P-P}, as described in [36]:

$$V_{P-P} = \frac{\Delta Q}{C_O} = \frac{1/2\left(T_{ON}\frac{I_{pk}}{4} + T_{OFF}\frac{I_{pk}}{4}\right)}{C_O} = \frac{I_{pk}\frac{T_{ON}}{T_{OFF}}}{8 \cdot C_O}$$

Please note that with higher C_O leakage increases through capacitor equivalent series resistance (ESR). Thus C_O is typically chosen to fit an expected maximum I_{pk} load. Another drawback is that if C_O is not sufficient, the consequence of going beyond its limits is higher, since the supplied current through I_L is also limited, and first has to charge C_O before the circuit is supplied.

2.3. Mixed-Signal Integrated Circuits

Many integrated circuits nowadays are not pure digital or pure analog, but typically contain subsystems of both types of circuits. The more applications a single chip has to support, the more likely that analog and digital blocks are integrated together. One of the biggest challenges in mixed-signal design is the noise susceptibility of the analog subsystem, which gets affected by the higher-frequency and higher-power digital subsystem.

If the digital logic consumes a high current as mentioned in Section 2.1, a voltage drop in the power supply becomes visible through slightly reduced supply voltage, which is also biasing analog components [37]. For very high frequency current changes, this voltage drop might just be observable inside the chip, with voltage fluctuations traveling through the chip-internal power supply mesh [16], or the common substrate of the whole die [38].

An additional effect to voltage fluctuations is chip-internal cross-talk from electromagnetic (EM) coupling. Depending on the frequency of a signal pulsed on a wire, the wire acts as a strong or weak radio transmitter, which also affects nearby wires, biasing wire delays through inductive or capacitive coupling effects [39]. Digital circuits are designed with a specific noise margin to prevent bit flips during normal operation, but analog circuits can be biased through EM [40].

In summary, it is usually hard to guarantee that digital circuits have zero effect on an adjacent analog circuit. Instead, a mixed-signal chip is designed such that the noise margin is considered sufficient for the application requirements. We will show that security requirements may impose much higher restrictions on the allowed noise levels, at least regarding the noise caused by digital components.

2.4. Analog-to-Digital Converters (ADCs)

Various types of an Analog-to-Digital Converter (ADC) are implemented in integrated circuits [41]. One of the most common and cheaper general-purpose designs is a *Successive Approximation Register* (SAR) ADC, which is also the type of ADC utilized in the systems evaluated later in this work, in Section 5.3. We discuss here to which extent ADCs can be influenced by digital noise in mixed-signal chips.

Classical ADC noise characterization analyzes the *effective number of bits* (ENOB) of an ADC over its actual number of bits [41]. This effective dynamic range depends on a number of distortion parameters, such as the signal-to-noise ratio (SNR) and spurious-free dynamic range (SFDR). These parameters are typically characterized and tested for the ADC circuit itself, and do not specifically involve noise from the digital subsystem of a mixed-signal circuit [42, 43].

SAR ADCs use multiple discrete timesteps over which the conversion is performed, where each timestep resembles one of the output bits of the ADC, i.e. a 12-bit ADC requires at least 12 ADC-internal clock cycles. In each of the timesteps, the current ADC input value is evaluated against a specific voltage level connected to a shared analog comparator. If we assume chip-internal noise affects a fixed voltage in a measurement result, the Least Significant Bits (LSBs) get affected more than the Most Significant Bits (MSBs). Thus, typically only during the conversion of the lower bits, a SAR ADC is susceptible to chip-internal noise.

Another ADC type that is often integrated in Systems on Chip (SoCs) or microcontrollers is a Sigma-Delta-ADC (ΣΔ-ADC), which also operates over multiple clock cycles. They perform a sort of approximation over multiple clock cycles. In each clock cycle, the difference of a previously measured value is compared with the input value and is integrated over multiple cycles. This integrated value can be affected by chip-internal noise on any of the integration clock cycles, and affect the LSBs. Through the nature of differential sampling, some of the noise can be rejected. However, noise affects a larger time window of the ADC result than for SAR ADCs, i.e. not just during measurement of the LSBs. So overall, we assume chip-internal noise can affect $\Sigma\Delta$-ADCs about the same as SAR ADCs.

2.5. Sensing and Protecting against On-Chip Voltage Fluctuations

Multiple works have addressed detecting process, voltage, and temperature (PVT) fluctuations which come from operating conditions, manufacturing variance or deliberate attacks. All of these fluctuations will affect path delays, where voltage variation can change as fast as within nanoseconds [44]. Some works concentrated on sensing errors caused by timing violations such as the 'Razor' architecture [45], also adapted to FPGAs [46–48]. After an error is detected, a rollback or similar strategy can recover the situation to guarantee further valid operation.

Other approaches monitor delays of surrogate structures that correlate with critical path delay. Such sensors can facilitate the analysis of all internal changes that cannot be seen externally, and include all potential side effects of a full system. Combinational delays can be self-measured in FPGAs [49], and steady-state process, voltage and temperature (PVT) variations have been characterized with Ring Oscillators (ROs) [50]. The measurement through ROs requires a counting mechanism. Because of that, even with multiple ROs, a sampling rate of only $8MHz$ was achieved in a technology operating at up to $600MHz$ circuit frequency, making them too slow to sense variations at circuit speed [51]. Yet, Time-to-Digital Converter (TDC) [52] sensors for accurate time-event measurement were already designed for FPGAs [53–55]. On ASIC systems, TDCs were adapted for path delay sensing from PVT variations [56–58]. Based on TDCs, Zick et al. [44] implemented a FPGA-based sensor to detect voltage undershoots, with the purpose of detecting voltage-based fault attack threats on security systems.

As we show in Figure 2.2, these sensors work by checking how far a signal (here: clock signal) can propagate through a path by adding latches between the logic elements of that path. The most simple path is a chain of buffers. As the delays of these buffers are sensitive to any joint PVT variations, they will become measurable by reading the latches. To do that, a signal is connected to both the input of the first buffer and the latch enable-signals. That means, the signal on the wire will be faster than the same signal delayed through the buffer elements. A readily available and precisely timed signal is the clock signal itself. At half of the clock period, the latches will become disabled and keep the state how far the clock propagated through the buffer chain, which can then be processed further. In the depicted case, the *Initial Delay* needs to be adjusted for less than a half of the clock period (time when latches are enabled). Then, latches are connected between the last buffer elements, marked as the *Observable Delay Line*, that falls within the expected delay variation that we want to observe with the sensor. This limited operating range is chosen, based on acceptable area overhead and expected operating range values. More detailed descriptions can be found in the aforementioned references.

Although different works consider voltage-induced delay fluctuation, they are either too slow to sense any fast transients [50], or were not yet further characterized to sense actual delay [44]. Additionally, post-processing is needed, in case e.g. binary values are required for further in-circuit processing.

For FPGAs, other primitives are used to resemble the buffers. In Xilinx 6 and 7-series FPGAs we use a 'CARRY4' primitive for each 4 buffers of the *Observable Delay Line* [44]. Thus, in an estimation of this sensor, for a Virtex-6 FPGA, one buffer equals 19.5ps in delay. To improve linearity of the output, we use a two-bit *bubble proof* priority encoder [59]. Due to this, the sensor used here can take on values in the range of $0 - 62$.

2.6. Traditional FPGA and Hardware Security Threats

The triad of information security goals consist of *Confidentiality*, *Integrity* and *Availability*. In the following, a short overview of traditional hardware security threats is given that affect these goals with the focus on FPGAs. The non-traditional threats which have been introduced by this thesis are later discussed in Chapter 6.

Confidentiality demands to protect access to secret information. In an FPGA, secret data can be deduced by monitoring and analyzing power, voltage, temperature or other emanations that can leak cryptographic keys, found through cryptanalysis [60–62]. For ASICs, hardware trojans can be inserted by a contracted manufacturer [63], where on FPGAs they can additionally be injected into existing bitstreams later [64]. *Integrity* describes maintaining data such that it can only be modified in an authorized way. It can be compromised among other techniques with electromagnetic radiation or power supply manipulation to cause bit flips from timing faults or soft errors [65, 66]. *Availability* is the goal to keep systems at service when required, and is threatened by DoS attacks that crash or destroy FPGAs. A way to overload an FPGA can be by excessive heat, generated by synthesizing ROs in great numbers. Overheating could destroy early-generation FPGAs [67]. In modern FPGAs, overheat protection is commonly available, where after a forced cooldown, the usage of the chip can be resumed.

Most of these attacks require physical access to the system. An exception is overheating which is on electrical level but triggered through bitstream access or partial reconfiguration. To investigate these threats, FPGA bitstream encryption and cracking

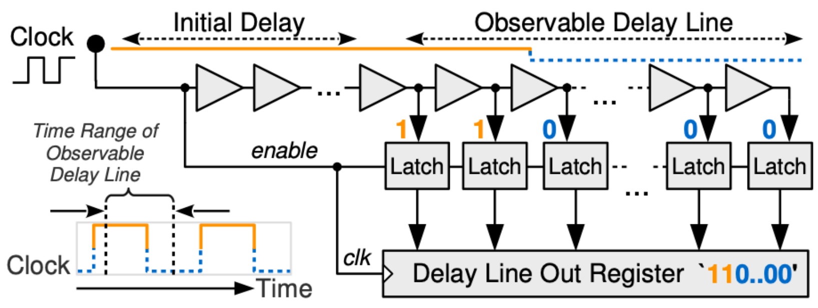


Figure 2.2.: Principle of a delay-chain based Time-to-Digital Converter (TDC), measuring the propagation of the clock signal.

are explored widely [68–71]. Unfortunately, guarding bitstreams is only a solution for a trusted software and hardware development approach, where bitstreams need to be signed or encrypted before the system accepts them. As reconfigurable systems and user-defined custom accelerators are becoming more widely adopted in FPGA systems, this fully trusted development approach becomes infeasible.

2.7. Leakage Assessment

In order to systematically evaluate exploitable leakage in any system, *leakage assessment* methodologies based on *non-specific fixed-vs-random t-testing* [72, 73] can be employed. Utilizing Welch's t-test for evaluating side-channel security of hardware implementations has been introduced in the seminal work by Goodwin et al. in 2011 [72]. The basic principle of all variants of this leakage assessment methodology is to test if statistical differences can be found in recorded side-channel data of the same cryptographic operation with different input values. Typically, two different sets of input data are chosen. For each set, side-channel traces are recorded, followed by an evaluation regarding their distinguishability.

Although a t-test evaluation does not allow an attacker to recover secret keys and break cryptographic implementations, this method is more generic, as it does not require to establish a hypothetical leakage model such as CPA [74]. It avoids the fixation on a specific intermediate value, such as a specific AES round. We briefly explain non-specific test-vector leakage assessment (TVLA), which can be used to evaluate basic side-channel attack vulnerability.

For a secret key encryption $\mathrm{Enc}(k, m)$ with secret key k and plaintext m, we choose a key k for all experiments and generate a set $M_R = \{m_{r1}, m_{r2}, ..., m_{rn}\}$ of random plaintexts m_{ri} as well as a single fixed plaintext m_{fixed}. Then, the tested platform alternatingly computes $\mathrm{Enc}(k, m_{\text{fixed}})$ and $\mathrm{Enc}(k, m_{ri}) \forall m_{ri} \in M_R$, while measurements of the same platform are performed. On the two sets of traces, the average, variance and higher order moments for every sample time step are computed [73]. The obtained values can be used to compute arbitrary order t-values for every sample time step as shown in Equation 2.1:

$$t = \frac{\mu_r - \mu_{fixed}}{\sqrt{\frac{s_r^2}{n_r} + \frac{s_{fixed}^2}{n_{fixed}}}} \tag{2.1}$$

In Equation 2.1, μ_r and μ_{fixed} are the raw averages of the two sets of traces during encryption of m_r and m_{fixed}, respectively, at a specific sample time step for a first order t-test or the higher order central moments for a higher order t-test. Likewise, s_r^2 and s_{fixed}^2 correspond to the respective variances for a first order t-test and the higher order standardized moments for a higher order t-test. The amounts of random and fixed traces are n_r and n_{fixed}.

To prove a design secure against side-channel attacks using leakage assessment, it is usually recommended to select multiple different fixed plaintexts and perform a leakage assessment for each of them, and record a significant amount of traces (i.e. $\geq$ 10 Million). As in this work we do not want to prove security but rather want to show information leakage, we typically evaluate a single fixed plaintext m_{fixed}, and record less than 10 Million traces.

Exploitable leakage is assumed for $|t| > 4.5$, a generally accepted threshold [72, 73]. A value of $|t| > 4.5$ relates to a confidence of > 0.99999 that the traces collected from random encryptions and those from fixed encryptions are samples drawn from different populations. When leakage assessment is performed in this work, sampling is synchronized with the beginning of the encryption algorithm and we restrict the leakage assessment to the middle third, as recommended in [72].

2.8. Correlation Power Analysis on AES

To recover secret keys of AES through power analysis, CPA with a leakage model is a well-known standard attack [74]. In order to recover a secret AES key, an attacker collects a certain amount of power traces to eventually find a distinct correlation between the collected traces and a power consumption model of the correct key candidate. These power traces are collected during AES encryptions, where either plaintexts or ciphertexts are known to the attacker. Attacks are usually performed on single key bytes of the first or last round key, where the attacker is able to compute power consumption models for all 2^8 possible key byte values in negligible time. The classical correlation model from [74] is based on the assumption that power consumption of computations depends on the Hamming distance between intermediate values, for instance after the *SubBytes* operation of AES:

$$P_{\text{hyp}} = \text{HW}(\text{SBox}^j(K_{\text{hyp}} \oplus S_i)) \tag{2.2}$$

In Equation 2.2, $\text{HW}(x)$ is the Hamming weight of x, $\text{SBox}^j(x)$ is the (inverted) *SubBytes* function of the AES algorithm and K_{hyp} is the key hypothesis byte. Depending on whether the first or the last round of the AES encryption is attacked, S_i is one byte from either the input plaintext or the output ciphertext, where $i \in 0, 1, ..., 15$. Moreover, SBox^j is the normal ($j = 1$) AES substitution function when the first round is attacked, and the inverted ($j = -1$) substitution when the last round is targeted.

Another possible Hamming weight model is based on the result of a T-table lookup. The AES algorithm can be optimized by implementing the *MixColumn* and *SubBytes* operations into a single table lookup, where each input byte yields a 32 bit output word. For CPA, the Hamming weight is then based on the output word of the T-table lookup function.

The CPA result for a single byte is based on computing the Pearson's correlation coefficient ρ_j between a hypothetical power model P_{hyp} and the actual measured value P_{trace_j} for every sampling step j, using all collected traces n:

$$\rho_j = \frac{n \cdot \sum P_{\text{hyp}} \cdot P_{\text{trace}_j} - \sum P_{\text{hyp}} \cdot \sum P_{\text{trace}_j}}{\sqrt{n \cdot \sum {P_{\text{trace}_j}}^2 - (\sum P_{\text{trace}_j})^2} \cdot \sqrt{n \cdot \sum {P_{\text{hyp}}}^2 - (\sum P_{\text{hyp}})^2}} \tag{2.3}$$

With a sufficient amount of traces, the correlation for the correct key byte value will eventually differ significantly at a specific sampling step from the correlations with the incorrect key byte values, allowing the attacker to determine the correct secret key byte.

A successful CPA depends on traces that are collected synchronous to the encryption algorithm. For that, an alignment to reduce synchronization inaccuracies can be performed. In Section 5.3 of this thesis, we use the following approach: We compute the total average trace over all collected traces and use a normalized cross-correlation based alignment algorithm. Each trace is shifted within a defined range and the normalized cross-correlation with the total average trace is computed as follows in Equation 2.4:

$$\rho_{cc}(s) = \frac{1}{\sigma \cdot \sigma_t} \sum_i (\mu_i - \mu) \cdot (t_{i+s} - \mu_t) \tag{2.4}$$

In the above equation, σ is the total standard deviation over all values, σ_t is the standard deviation over the current trace, μ_i is the total average at sampling point i, μ is the total average over all values, t_{i+s} is the current trace value at sampling point $i + s$ and μ_t is the total average of the current trace. The maximum cross-correlation value defines a new trace shifted by s, which is aligned with the total average.

2.9. Differential Fault Analysis on the AES

To be able to prove a successfully conducted on-chip fault attack on a cryptographic implementation, we look into a DFA method on an FPGA implementation of the block cipher AES. In this work, we attack an implementation with 128 bit key length. The encryption and decryption scheme is based on the circular application of four different operations *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey* on the data block, which is stored in a four by four byte matrix called the *state*. This circular application is repeated for 10 rounds as defined by the key length of 128 bits.

DFA is based on causing the same plaintext to be encrypted twice – the first time to gain the correct ciphertext and the second time to acquire a faulty ciphertext as the result of fault injection at a specific point of the algorithm. The ciphertext pairs, each consisting of a correct and a faulty ciphertext of the same plaintext, are then evaluated to extract information about the secret key of the cipher.

In 2003, a generic fault attack on Substitution-Permutation Network (SPN) ciphers was introduced, which only requires two ciphertext pairs to recover the original secret key [75]. We apply this attack with multiple adaptions for practical feasibility to use it on an AES module implemented on an FPGA, and therefore elaborate on this method in detail.

The fault model of the attack is a random byte fault on a single byte occurring before the 8th round of the AES algorithm. Before elaborating the eventual attack, which requires only two faulty ciphertext pairs to be successful, we consider a random byte fault before the 9th round. As depicted in the example in Figure 2.3, a single byte fault on the first byte of the state matrix before the 9th round (as well as on bytes 5, 10 or 15) is propagated and results in four faulty bytes at specific positions in the output ciphertext: 0, 7, 10 and 13. With a single byte fault on one of those bytes, we can therefore compute candidates for four bytes of the last round key. After recovering

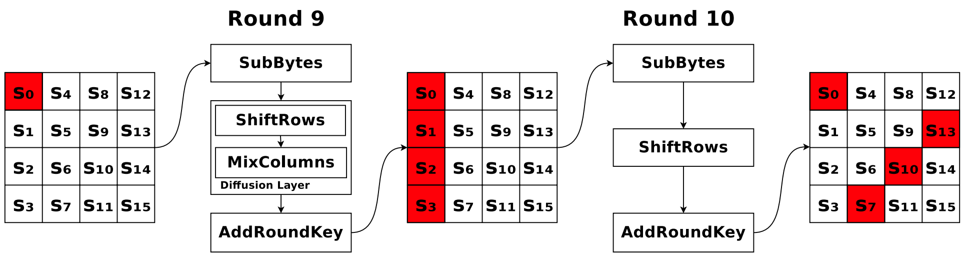


Figure 2.3.: Propagation of a faulty byte in the input of the 9th round of the AES algorithm

the entire 10th round key, it is possible to compute the original AES secret key, since the key schedule is invertible. Similar relations exist for the other bytes of the state matrix.

Exemplary, we consider single byte faults on the bytes 0, 5, 10 or 15 in the state matrix before round 9, which can be used to recover bytes 0, 7, 10 and 13 of the last round key. We initialize a set $\mathscr{S}$ containing all possible candidates for those bytes of the last round key. This set of possible candidates is continuously reduced with the evaluation of each pair of correct and faulty ciphertext (C, C') by inverting the 10th AES round for the two ciphertexts with a candidate $k \in \mathscr{S}$. The candidate is discarded, if the difference between the two state matrices resulting from the inversion of the ciphertexts is not within the set of possible differences resulting from a single byte fault on bytes 0, 5, 10 or 15 before round 9.

In [75], the number of candidates remaining after two ciphertext pairs was only one (the correct) candidate in 98% of all cases. In the other 2% of cases, only two or a maximum of four key candidates were left. Therefore, to recover the entire round key of round 10 with faults injected before round 9, a minimum of eight ciphertext pairs are required: For each of the four key bytes, two pairs are needed. This can be improved by injecting single byte faults before round 8, which affect four bytes before round 9 and therefore all bytes of the output at once. Then only two ciphertext pairs are required to recover the full AES key.

Part II.

Contributions

3. An Experimental Evaluation and Analysis of Transient Voltage Fluctuations in FPGAs

The work described in this chapter was published in [19] and is joint work with co-authors Fabian Oboril, Saman Kiamehr and Mehdi B. Tahoori. More details on contributions is found in Section 1.1.

A major threat to meet timing constraints in advanced technology nodes is the impact of process and runtime variabilities, which require the use of increasing timing margins [13, 14]. Among these variabilities, transient voltage fluctuation appears to have one of the most critical timing impacts [16, 17]. Additionally, they are elevated by more complex circuit designs that generate even more fluctuations originating from switching behavior, clock- or power-gating [15, 16]. As a result, conservative timing margins are too pessimistic and too costly, where countermeasures can be enabled by better analysis [15, 76].

Supply voltage fluctuations are caused by current drawn from the *Power Distribution Network* (PDN) due to switching activity and leakage power [16, 29]. This noise in the supply voltage level affects the timing of individual gates and is dependent on resistive (IR drop) and inductive (di/dt noise) components of the PDN [26, 28, 77]. From the temporal behavior on the circuit delay, it can be categorized as steady-state (mostly IR) and transient (IR+di/dt) voltage drop, with the second being more critical in advanced technology nodes.

Figure 3.1 shows our results of the relative timing impact (i.e. change in path delay) from different variations, which will be more detailed in Section 3.1.2. *Process variation*, *temperature difference of 30°C*, and *steady-state voltage drop* show averages over time with error bars from multiple locations on chip. For *transient voltage drop*, voltage undershoots are measured in time with 10ns resolution. Voltage drop is caused by toggling 8% of the flip-flops in the FPGA. Due to its high influence on path delay, these results highlight transient voltage drop as the most important problem.[1]

To improve margining, or perform design optimizations to handle such runtime variations, a deeper understanding of the impact of different on-chip activities, including both temporal and spatial behavior, is required. FPGAs as well as ASICs suffer from severe process and runtime variations in recent technology nodes. Moreover, typically only timing margins for a fixed frequency are applied in standard industrial FPGA mapping tools, and voltage or frequency scaling needs additional hardware and devel-

[1]Process variation can be handled by static margins, and we do not consider aging here, as this was already experimentally analyzed in [78].

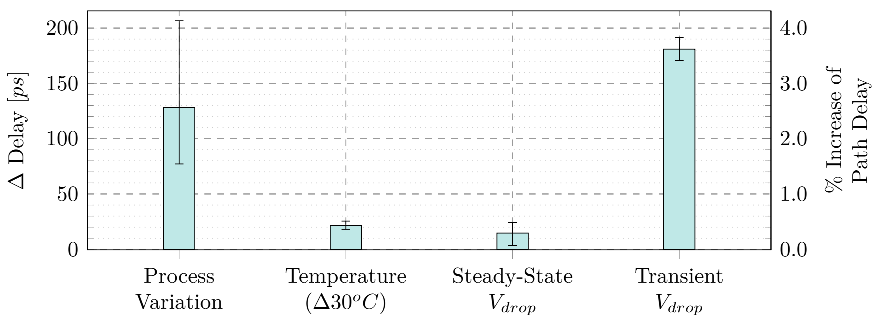


Figure 3.1.: Difference in delay from variation in manufacturing process, temperature and steady-state or transient V_{drop} (measured by different switching activity) of 8% of the flip-flops available in a Xilinx Virtex 6 XC6VLX240T FPGA, recorded across eight sensors. Percentage based on the idle sensor path delay at baseline temperature disregarding process variation.

opment effort. Besides that, FPGAs are very flexible to implement our experiments to analyze and understand runtime variations.

Traditionally, voltage stability has been analyzed on the level of PDNs for different ASICs and FPGAs [17, 29, 79–81], but did not contain insight on the influence from mapped circuits or workloads and the timing margins they would require. Furthermore, voltage regulators are typically excluded from these analyses. One post-fabrication analysis of high-performance processors looked into workloads at certain stimulus frequencies and whether they can be aligned in a beneficial way [82], but did not consider different workload runtimes. However, different transient behavior can have undesirable side effects, as seen in chip-testing [83–86], microprocessor-based systems [15, 16, 58, 87], and FPGAs, as shown in Figure 3.1. For FPGAs, process, voltage and temperature variations have been analyzed [50, 88], but without considering the transient effects of voltage fluctuations. Notably, one work shows the implementation of a TDC based on configurable logic that can sense fast timing transients [44], but is neither calibrated to absolute delay, nor used to analyze the influence from different switching activity.

In summary, more analysis considering the on-chip transient voltage drop in FPGAs is required. In this chapter, calibrated and redesigned TDC sensors to measure fast timing transients are presented. We design test structures and map them with multiple sensors into the FPGA, to study and investigate the impact of running workloads and mapped design on the transient voltage drop induced delay. This is analyzed under various workload durations (i.e. duty-cycles) and scheduling periods.[2] The results and findings are valuable to system designers by providing a chance to schedule and map workload for lowering the voltage drop, increasing reliability or performance.

The contributions of this chapter can be summarized as follows:

- Calibrating and analyzing a sensor for voltage drop induced path delay and preparing it for in-circuit use.

[2]These terms are defined and explained in Figure 3.5 in Section 3.2.

- Comparing the path delay impact of transient voltage drop with other process and runtime variations.
- Analyzing the impact of voltage drop on required timing margins caused by different switching activity, temporally and from various spatial activity patterns.
- Temporal and spatial quantification of voltage drop by analyzing the degree of attenuation or amplification from spatially and temporally distinct activities.
- The generality of the observed trends are verified on another FPGA chip manufactured on a more advanced technology node.

The remaining sections explain an adjusted TDC sensor including an initial analysis of process, voltage and temperature variation in Section 3.1. After that, Section 3.2 explains the experimental setup with detailed results. Section 3.3 concludes this chapter on voltage fluctuations.

3.1. Sensing Transient Delay Variation

For our evaluations, we need to sense both any subtle level of detail inside the PDN, as well as the spatial spread of voltage drop across the FPGA die. To measure these, the lowest cost approach is to implement sensors in the FPGA logic itself. This can be achieved by a TDC design and implementation based on the idea and related work discussed in Section 2.5. Additionally, the sensors are calibrated to sense absolute changes in delay.

3.1.1. Time-to-Digital Converter Implementation

As briefly mentioned in Section 2.5, to implement TDC on FPGA systems, we need to use the configurable fabric as a replacement for the buffers. We take advantage of previous experiments with TDCs [44, 52–54, 56, 59, 89] and use the same carry-chain primitives ‘CARRY4’ that are available throughout Xilinx FPGAs. The optimization of these primitives for a carry-chain allows for the smallest granularity between the bins of a TDC on a Xilinx Virtex 6 FPGA or comparable devices (i.e. also 7 series devices). Each CARRY4 consists of 4 configurable carry logic stages, where intermediate bins can be read. We use 16 of these elements, resulting in 64 quantization levels.

We utilized VHDL ‘generate’ statements to set Xilinx’ relative location constraints (RLOC) that are later evaluated in the mapping phase. That way we do not need to manually place and route the whole sensor. Using this strategy, sensor delay ranges can be explored faster to match the variations we want to observe. With directed routing constraints from *FPGA Editor* we can then verify the routing to be equal across the sensors, and if required adjust it. A floorplan of the final sensor’s delay line is shown on the right side of Figure 3.2.

Besides using VHDL, we extended previous work [44] with finer adjustments in the offset for the 64-value range, for which we partially use CARRY4 elements for the *Initial Delay*, which only consisted of ‘LD’ (Latch) and ‘LUT5’ (Lookup Table) primitives before. That is shown in Figure 2.2, in Section 2.5.

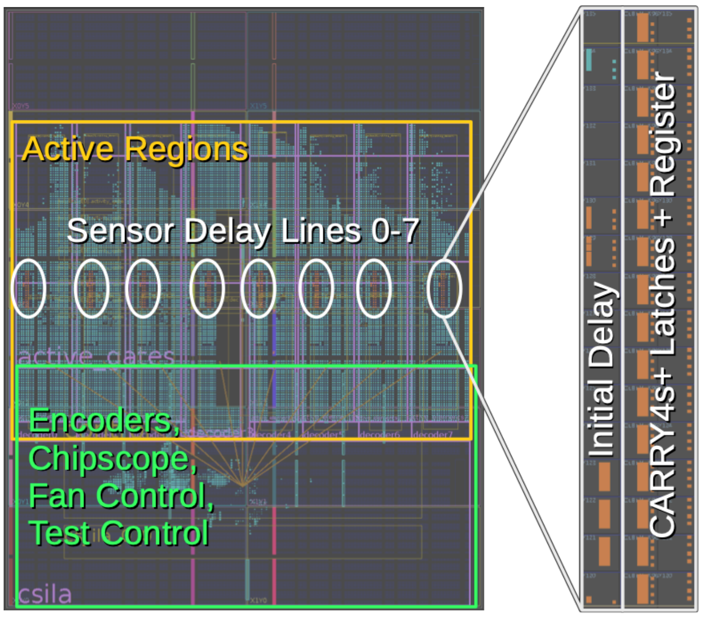

Active Regions
Sensor Delay Lines 0-7
active_gates
Encoders, Chipscope, Fan Control, Test Control
csila
Initial Delay
CARRY4s+ Latches + Register

Figure 3.2.: Floorplan of the Virtex-6 FPGA with 8 horizontal regions. A delay line including registers is shown magnified and annotated on the right side.

Additionally, we added a priority encoder to our sensor implementation that transforms the output of the delay line into a binary number, to make it usable for any live in-system evaluation using Xilinx Chipscope or mapped logic. This encoder was optimized for speed as described in [18].

When the sensor is sampled, the highest bit propagated through the buffer chain might not be followed by another '1'. Due to hold time noise of the latches, it can be followed by a '0'. In that case we discard the '1' as the highest, and check for the next '1' that was saved to the latches. In addition, some residual idle noise exists, which can be attributed to a non-ideal power supply, as analyzed in [18]. The total logic used for the sensor plus the encoder is shown in Table 3.1.

Table 3.1.: Resource use for one sensor, encoder, and registers between

Resource	Utilization		
(Xilinx Virtex 6)	Sensor	Encoder	Register
FLOP_LATCH	72	78	64
LUT	8	139	—
CARRY4	19	—	—
MUXFX (Multiplexer)	—	5	—
INV (Inverter)	—	1	—

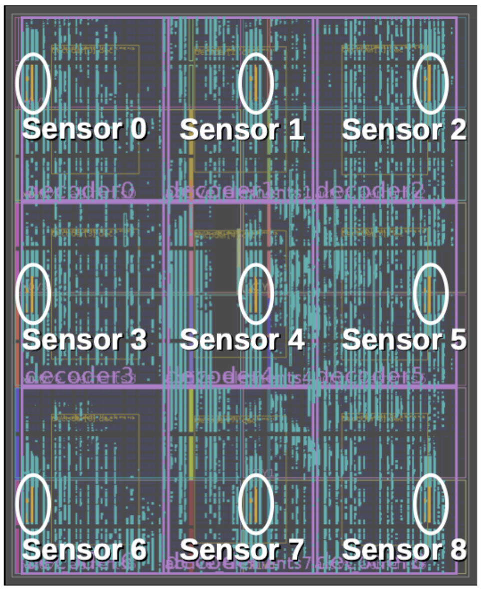


Figure 3.3.: Floorplan of the Virtex-6 FPGA with 9 sensors and regions.

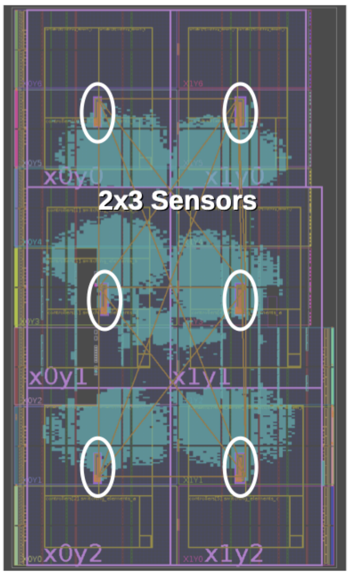


Figure 3.4.: Floorplan of the Kintex-7 to verify our results.

3.1.2. Sensor Calibration

The sensor delay line is sensitive to all variations that can affect the primitives used for it, i.e. process, voltage and temperature variation. Therefore to calibrate the sensors, we first need to analyze the impacts of these variability sources on the sensor readings, and separate them accordingly.

To differentiate among these variations, already shown in Figure 3.1, we have to evaluate the sensor under controlled conditions. Therefore, we place eight sensors horizontally across our FPGA, and read their output. For spatial *Process Variation*, we compare the idle delay of the sensors against each other, showing their minimum, maximum, and average difference, while the temperature was kept constantly at 33^oC for this observation. To show the delay variation from $\Delta 30^oC$, we first keep the system idle at 33^oC and then heat it up to 63^oC by switching activity (i.e., a self-heating mechanism), that we disable again for the measurement. To know the temperature, we check the built-in calibrated temperature sensor, for which we use only four of the delay sensors that are closer to this sensor in the center region. We show their minimum and maximum differences in that temperature range. For the *Steady-State* V_{drop}, we configure 8% of the FPGA's flip-flops and keep them toggling at $100MHz$, each connected to an inverter in a loop. We then compare the average sensor values with their idle averages. For *Transient* V_{drop}, we record multiple worst-case delay increases from voltage undershoots that occur from suddenly starting to toggle all the configured flip-flops at $100MHz$. The flip-flops are connected to a synchronous clock, but since the area over which they are distributed spans over multiple clock regions in

Table 3.2.: Calibration data for each sensor in layout with eight horizontal sensors: Time to bit 0 (i.e. initial delay) and average required delay per bit (linear estimate).

Time	Sensor							
	0	1	2	3	4	5	6	7
Time to Bit 0 [ns]	4.86	4.76	4.72	4.72	4.70	4.61	4.61	4.68
Average Delay/Bit [ps]	11.3	12.0	11.3	11.5	11.9	12.0	12.3	11.7

the FPGA, we expect that a significant clock skew is involved.

The results in Figure 3.1 show that transient voltage drops lead to the highest increase in delay, while process variations follow close. As our analysis is based on a Virtex 6 FPGA, fabricated using 40 nm process technology, we expect even higher variation for newer technology nodes [27].

For the rest of this work, we focus on the delay influence from transient voltage drop. We consequently introduce a temperature-controlled fan to reduce thermal variation to a minimum. For process variation, we calibrate the sensors at a defined temperature, achieved through this fan control. We perform a two-point calibration, where we have a *Gain* in Delay/Bit and *Offset* as the Time to Bit 0. This calibration is done by using different frequency around our nominal sensor sampling frequency of $100MHz$ (i.e. sampled every $10ns$), while the system is idle except that chipscope is active. For the first calibration point, the sensor is connected to $100MHz$. The FPGA integrated clock control (Xilinx Multi-Mode Clock Manager) allows to set $92.308MHz$ as the next lowest and $109.091MHz$ as the next highest frequency. These were used to acquire calibration values for all sensors. The clock signal reaches both the first buffer element and the latches connected to the observable delay (also see Figure 2.2) at the same time with an uncertainty of $43ps$ across the FPGA, so we expect much less within one sensor. Essentially, the sensor bits encode the time until the latches get disabled. This time is half the clock period, being $5.00ns$, $5.42ns$ and $4.58ns$ for $100.00MHz$, $92.31MHz$ and $109.09MHz$ respectively.

This calibration leads to $11.7ps$ per bit average delay and $4.7ns$ average time to bit 0 (i.e. initial delay) on one ML605 board, with similar results on another ML605 board, confirming the results of process variation shown in [50]. The Xilinx ISE software estimates $78ps$ for one 'CARRY4' element, and by that $19.5ps$/bit. This difference shows that still about 40% timing safety margins are left, when the system is idle at 38^oC used for this calibration, and underlines how conservative these margins are. We repeated similar calibration steps on a Kintex-7 KC705 board. For that platform, an average delay per bin of $11.5ps$ was found, which is very similar, despite the technology difference. On that platform, the tool estimate is only $13.25ps$/bit, showing the use of less safety margins of only about 15%. In all timing estimates, the slow '`-2`' model was used, matching the FPGAs. For one of our setups, the detailed per-sensor calibration values are shown in Table 3.2, including the time to bit 0 and the average delay per bit.

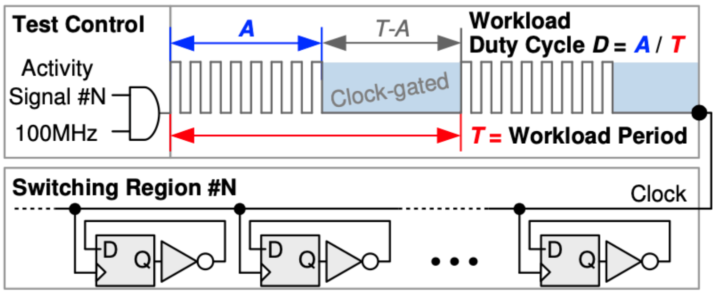


Figure 3.5.: Scheme of activating and deactivating workload through *Workload Duty Cycle* and *Workload Period*, activated from the Test Control structure, embedded in the test setup as shown in Figure 3.6. Flip-flops are reset to the same state after system startup.

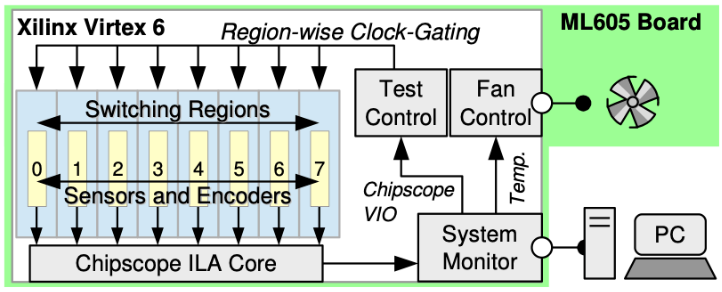


Figure 3.6.: Overview of experimental setup, showing connections of FPGA-internal and external blocks. Explanation of *Test Control* shown in Figure 3.5

3.2. Experimental Setup and Results

To evaluate the impact of different workload switching activities on transient voltage drop, we use various workload schemes, explained with the help of Figure 3.5. By using an *Activity Signal*, controlled from our *Test Control* Structure, we set various lengths of active or inactive times for a region of multiple flip-flops. During the active times A, the clock is toggling and the flip-flops are operating at a frequency of $100MHz$. During inactive times, a clock-gating signal deactivates the flip-flops. Using this concept, we can stimulate the system with a range of synthetic workloads.

We define and use a set of notations in this chapter, summarized in Table 3.3. *Workload Period* (T) is defined as the period in which active and inactive phases are alternating. Any switching region #N is enabled for the active time A and disabled in the remaining time $T - A$. From that, we define the ratio between the active time A over the total *Workload Period* time T as *Workload Duty Cycle* $D = A/T$. Additionally to that, multiple workloads can run at the same time, at which a *Workload Phase Shift* φ resembles their respective phase alignment, given in percentage % (i.e. $50\% \equiv 180°$). Thus, each workload w is defined through a set of three parameters $w = (T, D, \varphi)$. Figure 3.7 shows example activities to explain each parameter.

Table 3.3.: Notations describing workloads w and activity patterns W.

Notation	Description
T	Workload Period
D	Workload Duty Cycle, fraction of active time during T
φ	Workload Phase Shift, shift of start-time of workload in percentage of T
$w = (T, D, \varphi)$	Definition of a single workload w
$W = [w_1 w_2 .. w_N]$	Workload activity pattern with one workload for each 1..N regions
$0 = (0, 0, 0)$	Workload that is completely inactive

Table 3.4.: Overview of workloads w used in W in this chapter.

w	T	D	φ	w	T	D	φ
g	12.8 μs	50%	0%	m	51.2 μs	50%	0%
h	12.8 μs	50%	**50%**	n	51.2 μs	20%	0%
i	102.4 μs	20%	0%	o	51.2 μs	80%	20%
j	102.4 μs	50%	0%	p	51.2 μs	80%	30%
k	102.4 μs	50%	**10%**	q	51.2 μs	50%	50%
l	102.4 μs	50%	**50%**	0	N/A	0%	N/A

As a notation to show the activity of a full FPGA, we define *Workload Activity Patterns* (W), which resemble workloads at different regions of the FPGA. For eight horizontal regions this can be a string $W = w_1 w_2 ... w_8$ for workloads 1 to 8. For regions in two dimensions in a 3-by-3 layout, it is a matrix containing workloads in x and y dimensions w_{xy}, that can also be shown in a table format:

$$W = \begin{bmatrix} w_{02} & w_{12} & w_{22} \\ w_{01} & w_{11} & w_{21} \\ w_{00} & w_{10} & w_{20} \end{bmatrix} = \underset{x \rightarrow}{y \uparrow} \begin{array}{|c|c|c|} \hline w_{02} & w_{12} & w_{22} \\ \hline w_{01} & w_{11} & w_{21} \\ \hline w_{00} & w_{10} & w_{20} \\ \hline \end{array}$$

The workload '0' is defined as a placeholder that shows that no workload is active at that specific region. In Table 3.4 we list all the workloads used in any W in this chapter.

We design four experiments to analyze various characteristics of spatio-temporal voltage drop. Workloads are placed at different locations across the FPGA, operating at various parameters for T, D, φ. These are the following experiments:

Exp. 1: Analyze voltage drop depending on the impact of workload period and workload duty cycle.

Exp. 2: Evaluate the spatial spread of voltage drop across the FPGA from a single workload.

Exp. 3: Measure the voltage drop from temporal and spatial interference between multiple workloads on the FPGA.

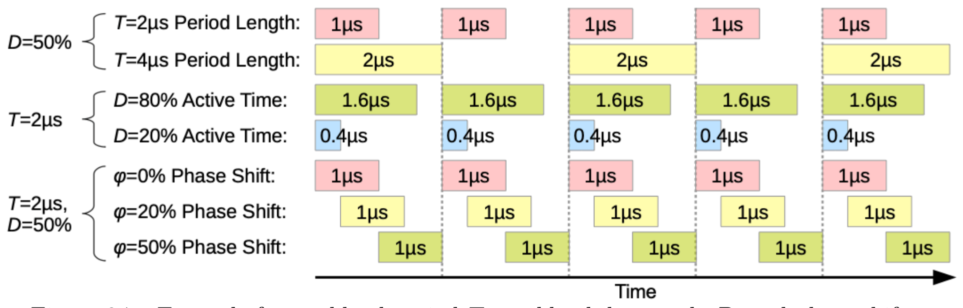


Figure 3.7.: Example for workload period T, workload duty cycle D, and phase shift φ

Exp. 4: Extend upon Exp. 3 by analyzing spatial interference in two dimensions across the FPGA fabric.

The overview of the system used for these experiments is shown in Figure 3.6, which shows a configuration with 8 sensors. For all experiments, we place multiple sensors across a Virtex 6 FPGA on a ML605 REV E Evaluation Board. Some experiments are cross-checked on a Kintex-7 KC705 board.

Both boards are supplied by a 12V AC/DC power supply on board-level, and use multiple DC/DC switched-mode VRMs for different voltage levels [90]. The ML605 FPGA-internal VCCINT voltage is generated with a Texas Instruments UCD9240PFC Digital PWM System Controller and PTD08A020W power conversion module [91], where we could measure 500 kHz as the PWM frequency. The KC705 power supply is based on the Texas Instruments UCD9248PFC.

For the first three experiments, we exclusively use 8 sensors placed horizontally from the left to the right side of the FPGA, later we use 9 sensors in a 3x3 layout. Each sensor is embedded in a region consisting of the same number of flip-flops connected to inverters, as shown in Figure 3.5. The floorplans of our test setups are shown in Figure 3.2 and Figure 3.3. The test structure, when 8 horizontally aligned regions are used, can be seen in Figure 3.2, including a magnification of one delay line with registers. In that setup, 14% of the available flip-flops on the FPGA are configured. A similar setup is available with only 8% flip-flops. Figure 3.3 shows a setup with 9 regions in which again 14% of available flip flops are configured (1.6% per region). In the following experiments, parts or all of these flip-flops toggle simultaneously, according to the scheme in Figure 3.5. In most experiments, less than 8% of the flip-flops are toggling in total.

This value is chosen, as it still reflects a realistic possible amount of toggling flip-flops of a circuit (as we have commonly seen in simulations of benchmark circuits such as a Leon3 processor), and leads to fluctuations that still fit inside the observable delay range of the sensors, as we found empirically. To make a trade-off between development effort and the time needed to acquire results, we designed a semi-automatic approach with a simple state machine and on-board control, with PC-based data acquisition.

3.2.1. System and Measurement Control

To transfer the sensor data to a PC, we use Xilinx Chipscope for data acquisition. Chipscope is an on-chip logic analyzer that takes advantage of the configurable logic. Typically, any signal in the FPGA can be connected and sampled at circuit speed and saved in on-chip block RAM as sample-memory, which can then be transferred by JTAG to the PC. To minimize the influence on the results, we limited the sample buffer to 32768 time samples of each $10ns$. For each sample we show the value for all sensors and informational signals that show the workload activity. As the Block RAM (BRAM) is distributed across the FPGA, and to minimize influence on the switching areas, we only used the resources available in the lower regions. We use *area groups* to constrain the major part of the used control logic to areas farther away from the sensors, reducing their influence on the measurements.

The experiments can be set to different modes by Chipscope Virtual Input/Output (VIO) to control workload periods, duty cycles and active regions. Additional to Chipscope, we added a simple fan regulator to concentrate the experiments on transient voltage sensing with a calibration for process variation as shown in Section 3.1.2 before. We verified that the fan controller has negligible impact on the results.

To reduce measurement variance, we record the maximum delay increases from at least 10 workload periods. Therefore, for longer periods, we save multiple full traces of 32768 samples ($327.68\mu s$). The principle setup of the connections among the blocks is shown in Figure 3.6. Test Control sets region-wise clock-gating at different duty cycles and frequencies. Xilinx' on-chip *System Monitor* is used to provide a temperature reading to the fan control and connects the Chipscope ILA core to the PC for data acquisition and VIO to define parameters for our Test Control block. The floorplan of the chip, together with the floorplan of a single sensor, is shown in Figure 3.2.

3.2.2. Experiment 1: Duty Cycle and Frequencies

The purpose of the first experiment is to study the effect of workload switching activity behaviors on transient voltage drop, and in particular the increase in delay from a voltage undershoot. For the first experiment, we use one large switching region, where in total 8.2% of the FPGA flip-flops are configured and connected to inverters, equally distributed across the center and switching at $100MHz$ when being enabled. However, the trends of the following results were also verified with $50MHz$, and we also cross-check them with $200MHz$ on a Kintex-7 KC705 board.

Three different time series of the delay read out from our sensors are shown in Figure 3.8. Each of them shows a case that continuously operates according to a scheme as generated by *Test Control* (Figure 3.5). In Figure 3.8 A (90%), we show the delay for a workload period of $102.4\mu s$ and 90% workload duty cycle. A small increase in delay is visible over the average ($\Delta Delay = 0ps$) during the active time, while for 10% of the time (when the workload is inactive), the delay is lower than at idle (negative $\Delta Delay$ values). The worst case delay over idle is $107ps$. On the other hand, Figure 3.8 B shows when only 10% of the time the flip-flops are active, resulting in a more critical outcome: Their worst case delay can reach almost twice of the previous case ($195ps$ vs. $107ps$) during a shorter time of activity. The third case, Figure 3.8 C

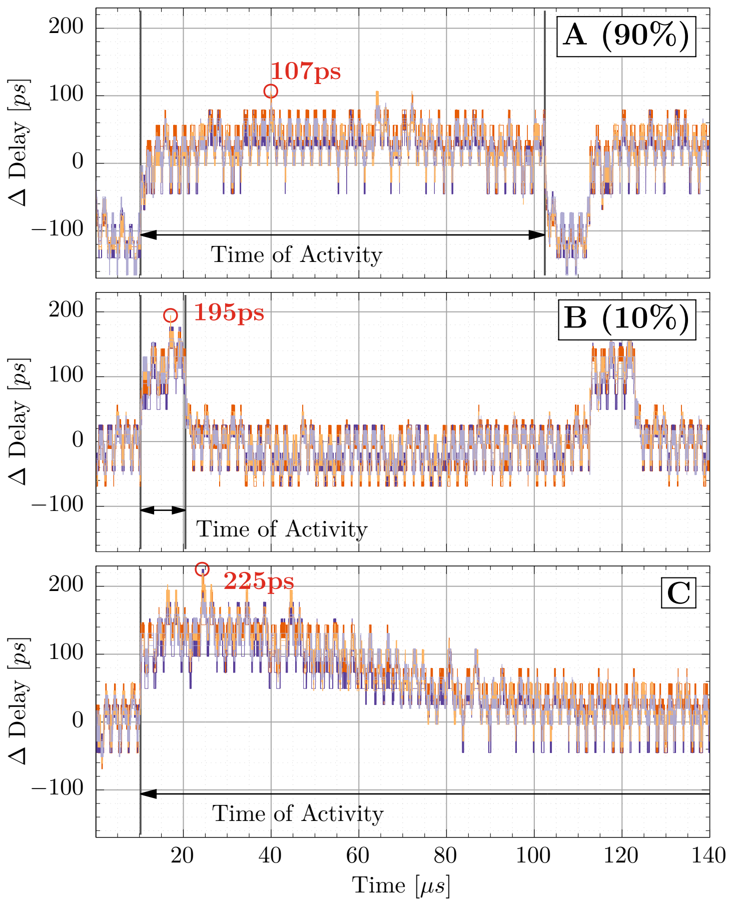


Figure 3.8.: Plot of all eight sensors each, while 8% of the FPGA's flip-flops toggle at 100MHz when being active, or are clock-gated/inactive. **A 90% / B 10%:** 102.4μs workload period, with 90% / 10% duty cycle, **C:** Starting switching activity after several seconds of inactivity. **Y-Axis:** Delay difference to idle, **X-Axis:** Time

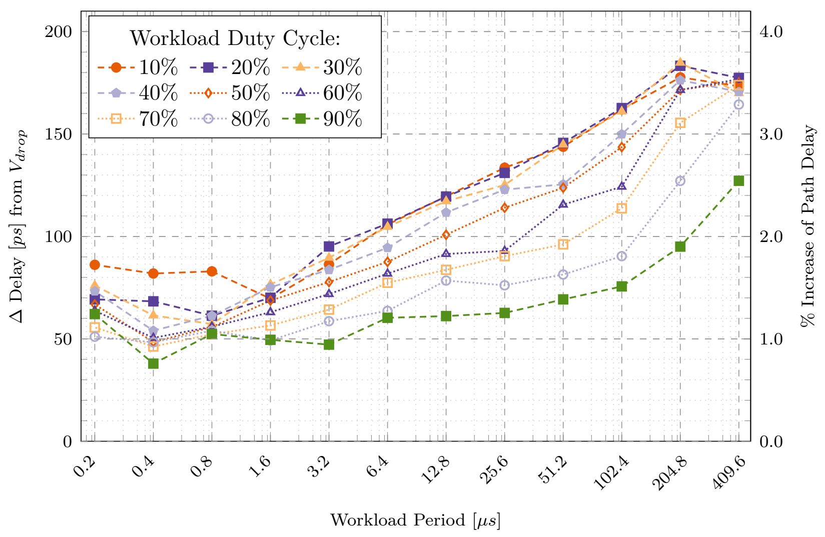


Figure 3.9.: Average of multiple worst-case delay increases from switching activity-dependent voltage drop, averaged across all sensors. **X-Axis:** Workload period length during the activity. **Y-Axis:** Delay difference to idle.

shows a corner case when the flip-flops start switching after several seconds of inactivity, resembling a step response to a sudden increase in current. The delay reaches up to a worst point of $225ps$, even worse than the $195ps$ of case B (Figure 3.8 B).

We repeat these experiments for several combinations of workload duty cycles (10-90%) and logarithmic increasing workload periods ($200ns$–$409.6\mu s$), and summarize the average of multiple worst case delay increases of multiple sensors in Figure 3.9. Please note, each workload duty cycle (10..90%) relates to a total overall switching activity at that percentage of time, with only shorter or longer workload periods i.e. alternating between active or inactive (as previously described using Figure 3.5 in Section 3.2). These results show the same trend of worst-case delay dependency on duty cycle as seen in the time series diagrams in Figure 3.8. For the observed workload period range in our system, we can conclude the following observations:

- Higher workload duty cycles typically lead to lower delay increase due to transient voltage drop.
- Higher workload periods lead to higher delay increase due to transient voltage drop.

Intuitively, the di/dt component when the flip-flops start toggling should be the same for any duty cycle. The IR drop component would be higher, due to overall more switching activity. This explanation makes our observations counter-intuitive, as we see higher delay increases from shorter duty cycles. Moreover, as shown in Figure 3.8 C, some time passes until the worst increase in delay is reached.

To explain these counter-intuitive results, we show the impact of duty cycle and workload period on the average delay (i.e. steady-state) instead of maximum delay.

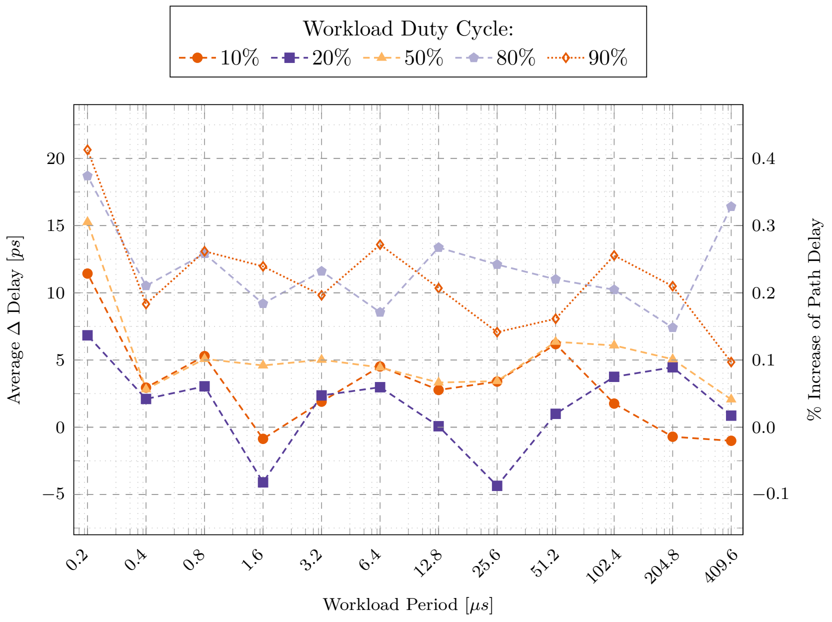


Figure 3.10.: Overall average delay at different switching activity, showing all relevant workload duty cycles (the values of the remaining duty cycles are within the boundaries of the shown duty cycles) **X-Axis:** Workload period length during the activity. **Y-Axis:** Delay difference to idle.

These results are depicted in Figure 3.10. In that diagram, only minor differences among different workload duty cycles can be seen. These additional results can be explained as follows: The on-board switched-mode VRM (cf. Section 2.2) is regulating for a defined target voltage, but as its regulation loop is typically working at a much lower frequency than the circuit behavior (depending on the workload frequency $= 1/T$, and the circuit frequency $= 100MHz$), any smaller changes are seen as a contribution to the *average consumed energy*, depleted from its output capacitor C_O (cf. Figure 2.1), for which the regulator stabilizes. Thus, the regulator will supply less current after intervals with less energy consumption (low workload duty cycle), than after high energy consumption (high workload duty cycle). Less current means shorter T_{ON} times of the regulator output, leading to higher voltage ripple V_{P-P} (cf. Section 2.2). As a result, the shorter the time a certain activity contributes to the average, the less weighted it will be, leading to higher transient voltage drop and delay increase.

To validate the generality of the observations made, we repeated some of the experiments on a Kintex-7 KC705 evaluation board (28nm FPGA), and show them in Figure 3.11. We add one more lower duty cycle of $0.1\mu s$, and also use flip-flops switching at 200MHz instead of 100MHz. Despite the changes, a trend of lower duty cycles leading to higher voltage drop is still visible in a major part of the analyzed workload periods.

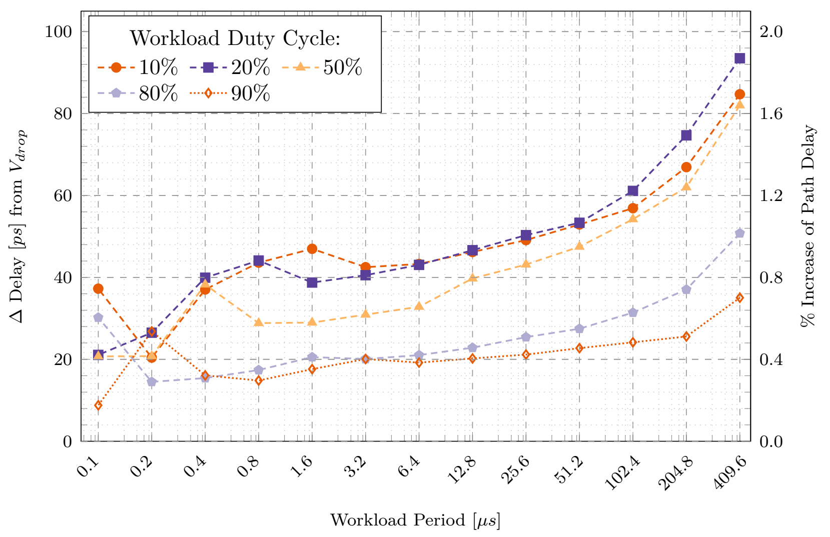
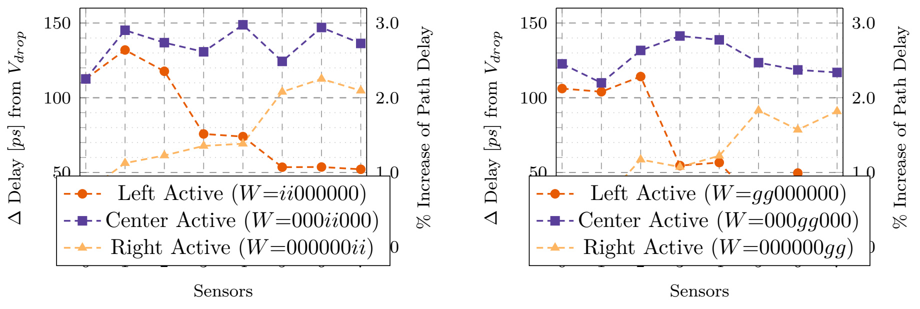


Figure 3.11.: **Cross-check on the Kintex-7 KC705 (cf. Floorplan in Figure 3.4), with Flip-Flops switching at 200MHz instead of 100MHz:** Average of multiple worst-case delay increases from switching activity-dependent voltage drop, averaged across all sensors. **X-Axis:** Workload period length during the activity. **Y-Axis:** Delay difference to idle.

3.2.3. Experiment 2: Spatial Spread of Voltage Drop

In ASIC designs, floorplanning decisions can involve a PDN optimization, tailored to the typical workload switching conditions of the circuit [17]. In FPGAs, circuits and workloads can be mapped in any way, which is unknown at fabrication time. This freedom leads to over-conservatively fabricated PDNs and timing margins applied during the mapping of a design. Hence, it gives an opportunity to exploit these margins by doing mapping and workload scheduling based on expected voltage variations. To this end, in addition to the analysis results, we obtained in the first experiment, we also need to analyze the spatial impact of transient voltage drops across the FPGA, i.e. the extent of delay increase in other regions as a function of the distance from the sources of transients (highly active regions) on the chip.

We configure in total 14% of the FPGAs flip-flops, which are distributed in eight regions with an equal amount of flip-flops. Each region is located around one sensor, and can be clock-gated separately. Two of these regions are enabled at the left, center, or right of the FPGA simultaneously, which is 3.5% of the total available flip-flops in each case. We test a variation of workload duty cycles and periods. However, the resulting spatial gradient showed the same trends and a similar difference in overall magnitude like in Experiment 1. This can be seen from the results for workload $i = (102.4\mu s, 20\%, 0\%)$ and $g = (12.8\mu s, 50\%, 0\%)$, in Figure 3.12. The important observations are:



(a) Spatial Differences with $T = 102.4\mu s, D = 20\%$

(b) Spatial Differences with $T = 12.8\mu s, D = 50\%$

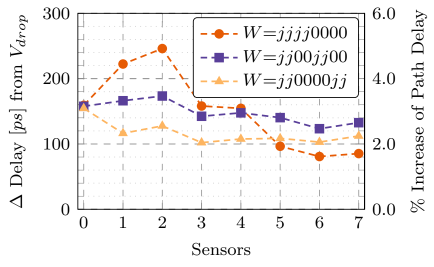
Figure 3.12.: Average of multiple worst-case delay increases from voltage drop per sensor, depending on activity in two regions (3.5% of total flip-flops), with two different D and T, $i = (102.4\mu s, 20\%, 0\%)$ and $g = (12.8\mu s, 50\%, 0\%)$. **X-Axis:** Sensors across the FPGA from Left(0) to Right(7). **Y-Axis:** Delay difference to idle

- An obvious difference in delay increase from inactive to active regions of up to 2.5× is visible.
- When the left or right regions are active, the gradient shape from either edge to the other edge shows an almost symmetric profile, potentially making a fitted function applicable to the whole FPGA.
- Although the shape of the curves is similar, the activity of the same number of flip-flops on the right chip side leads to an overall lower drop. Mis-calibration can account for a part of that, but at least the center two sensors would show the same offset at the same power. As they are different, we assume the right side of the chip is operating at lower power, which is reasonable according to [50].
- Activating the two regions in the *Center* shows the worst effect on the delay overall, as even the right and left side are affected more by activity in the center than their own region. The Xilinx' *System Monitor* logic in the center needs power as well and could possibly influence this. In addition, the center region has a higher distance to power/ground I/O pins.

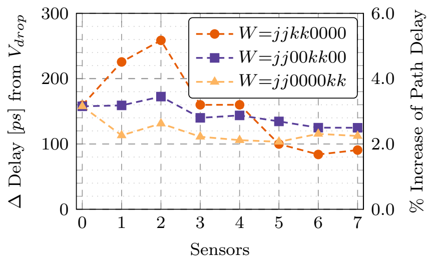
3.2.4. Experiment 3: Spatial and Temporal Interference among Workloads in Horizontal Regions

Usually, during realistic mapping and runtime decisions, more than one type of workload has to be active. Thus, in addition to the previous experiments, we analyze the interference among various simultaneously mapped workloads on the FPGA to observe whether they attenuate or aggravate the resulting voltage drops.

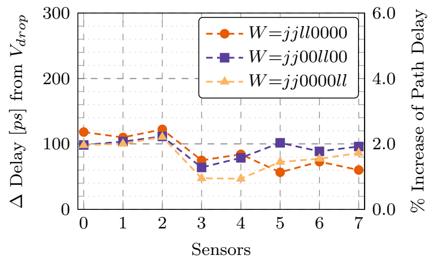
In this subsection, we show different workload activity patterns $W = w_1w_2..w_8$ based on the previous setup of 8 horizontal regions of which each uses 1.75% of the total



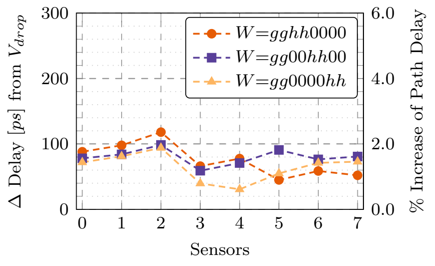
(a) $T = 102.4\mu s$, no phase shifts



(b) $T = 102.4\mu s$, with $\varphi = 10\%$ in workload k



(c) $T = 102.4\mu s$, with $\varphi = 50\%$ in workload l



(d) $T = 12.8\mu s$, with $\varphi = 50\%$ in workload h

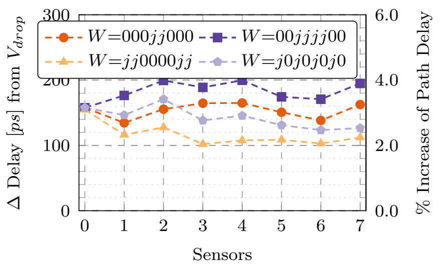
Figure 3.13.: Average of multiple worst-case delay increases from voltage drop per sensor, depending on activity with different phase shifts at different regions. Workloads summarized in Table 3.4. **X-Axis:** Sensors across the FPGA from Left(0) to Right(7). **Y-Axis:** Delay difference to idle

available flip-flops in the FPGA. Different effects will be explained based on selected subsets of our results.

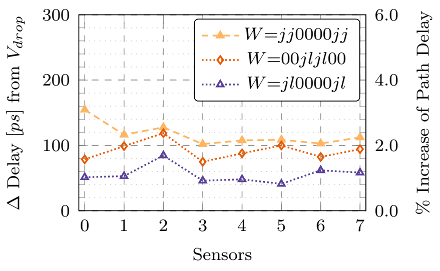
One general observation is that among different workloads, a high phase shift and a high spatial distance reduces the voltage drop. We provide evidence for this observation with the results in Figure 3.13. Three diagrams (a-c) show the effect of workload activity patterns that use workloads with either no (j), only $\varphi = 10\%$ (k) or the maximum $\varphi = 50\%$ (l) phase shift.

Without any phase shift, shown in 3.13a, the highest voltage drop (i.e., maximum amplification) occurs when the active regions (with workload j) are close together on one side as $W = jjjj0000$. In contrast, the voltage drop is reduced more (i.e., more attenuation) when the active regions are more apart ($W = jj0000jj$). When we apply a minor phase shift of $\varphi = 10\%$ to half of the active workloads (workloads k instead of j), shown in Figure 3.13b, there is no major change to that. However, the voltage drop reduces significantly when half of the workloads use $\varphi = 50\%$ for workload l, as visualized in Figure 3.13c. The maximum attenuation of voltage drop occurs when both high spatial distance and high phase shift are applied ($W = jj0000ll$). We also verify this trend for a different workload period $T = 12.8\mu s$ in Figure 3.13d.

In Figure 3.12 we provided evidence that the same activity can lead to stronger or weaker voltage drop, depending on its location in the system. An active workload in the center leads to a system-wide higher voltage drop than the same activity at an edge. This nonuniform spatial influence has to be considered by any approach that



(a) $T = 102.4\mu s$, no phase shifts



(b) $T = 102.4\mu s$, with $\varphi = 50\%$ in workload l

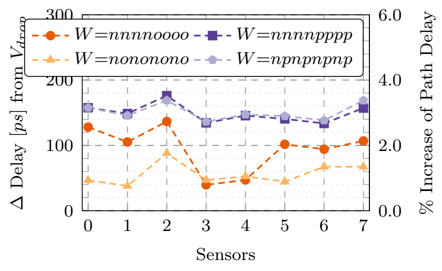
Figure 3.14.: Average of multiple worst-case delay increases from voltage drop per sensor. These workload activity patterns show the benefits of high phase shift and spatial distance among workloads. Workloads are summarized in Table 3.4. **X-Axis:** Sensors across the FPGA from Left(0) to Right(7). **Y-Axis:** Delay difference to idle

optimizes the placement of workloads for reduced voltage drop.

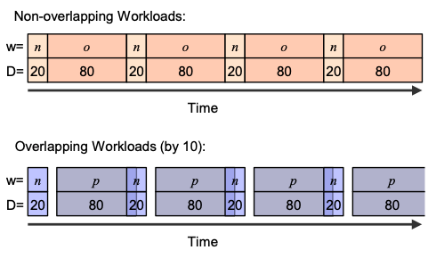
In Figure 3.14 we show two diagrams with different W patterns to compare the possible voltage drop reduction through phase shifting with that of spatial spread. The left diagram, Figure 3.14a shows that workloads in the center lead to the highest delay increase with the pattern $W = 00jjjj00$. A pattern which only has workloads on the two center regions ($W = 000jj000$) causes a higher voltage drop impact than even four of these workloads spread more evenly across the FPGA ($W = j0j0j0j0$). As one of the center regions is still active in this W, moving the four workloads completely to the edges ($W = jj0000jj$) will further reduce the voltage drop.

Figure 3.14b shows that phase shifting reduces voltage drop more than spatial spread. If applied to the center four regions as $W = 00jljl00$, phase-shifting is still more beneficial than just a spatial spread to the corners ($W = jj0000jj$), despite the center-region usually leading to higher voltage drop. Both spatial spread and phase shift combined ($W = jl0000jl$) result in the least delay increase, i.e., maximum voltage drop attenuation.

The results so far suggest that shifting an active time period with φ up to 50% always helps in reducing the voltage drop, by distributing the overall system workload more evenly in time. Yet, depending on other parameters, that is not always the case. In Figure 3.15a, we show a case in which a small increase in phase shift can lead to a high increase in voltage drop, instead of a decrease. A time diagram in Figure 3.15b explains that, by showing the active periods of workloads n, o and p and their potential overlapping times of activity. The least overlap exists with n and o, because of the combination of their duty cycles (20% and 80%) and phase shifts (0% and 20%) that make them non-overlapping. However, when the second phase shift of 20% is increased to 30% by replacing workload o with p, the resulting overlap leads to less distribution in time and an increased voltage drop. Similar interactions can be expected when workload periods or duty cycles are chosen in such a way to have overlapping times of activity.



(a) Variations of $n = (51.2\mu s, 20\%, 0)$ with $o = (51.2\mu s, 80\%, \mathbf{20\%})$ or $p = (51.2\mu s, 80\%, \mathbf{30\%})$



Non-overlapping Workloads:
W= n o n o n o n o
D= 20 80 20 80 20 80 20 80
Time
Overlapping Workloads (by 10):
W= n p n p n p n p
D= 20 80 20 80 20 80 20 80
Time

(b) View on workloads in time when either no overlap happens (between n and o) or with an overlap of n and p)

Figure 3.15.: Average of multiple worst-case delay increases from voltage drop per sensor. These workload activity patterns show the high impact a small phase shift can have. Workloads are summarized in Table 3.4. **X-Axis:** Sensors across the FPGA from Left(0) to Right(7). **Y-Axis:** Delay difference to idle

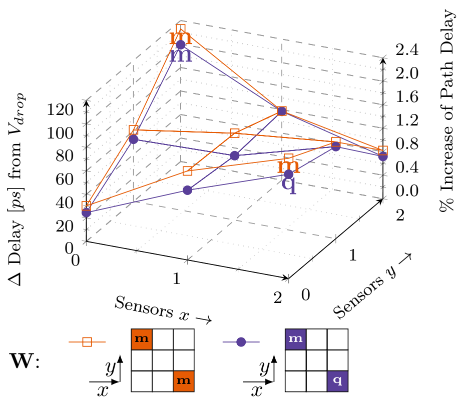
The important additional observations from this subsection, with respect to interdependence, are:

- The trend of spatial spread is not significantly influenced from changing only workload period or duty cycle.
- Changing duty cycle or workload is interdependent with phase shift in case of overlaps in time (cf. Figure 3.15).
- Distributing workloads more uniformly in time reduces voltage drop (i.e. through appropriate phase shifts).
- Distributing workloads spatially reduces voltage drop, however, nonuniform spatial influence should be taken into account (i.e. some regions lead to more voltage drop than others).

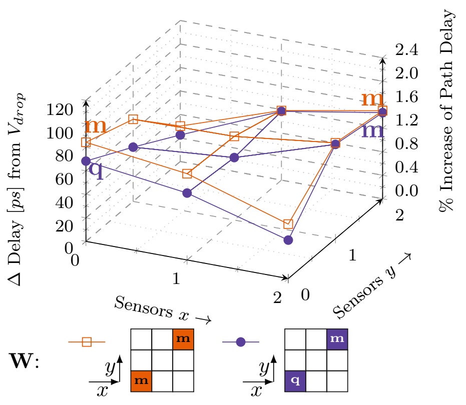
3.2.5. Experiment 4: Spatial and Temporal Interference among Workloads on 3x3 Regions

The previous experiments show detailed results horizontally across the system, presented through 2D diagrams, sufficient to catch and explain most interdependence in the system. In order to confirm these results and analyze additional effects both vertically and horizontally across the chip, the 1D setup is now extended to a 2D setup. For this, we show plots of a layout with 3x3 regions, where the floorplan was shown in Figure 3.3. In each of these regions 1.6% of the total FPGA flip-flops are configured and can be set to specific workloads.

From one corner to the other corner of the FPGA, phase shift will still attenuate voltage drop. Figure 3.16 shows two diagrams in which either of two corners is active with one of two workloads. Workload m is in phase with $\varphi = 0$, while workload q is shifted by $\varphi = 50\%$. As Figure 3.16a and Figure 3.16b show, for both pairs of



(a) Two corners being active without phase shift (—□—) and with 50% phase shift (—●—)



(b) Two other corners being active without phase shift (—□—) and with 50% phase shift (—●—)

Figure 3.16.: Average of multiple worst-case delay increases from voltage drop per sensor. These workload activity patterns show the effect of an activity in either of the two corners, with one activity 50% phase shifted (—●—) or without phase shift (—□—). Workloads are summarized in Table 3.4. **X/Y-Axis:** Sensors across the FPGA. **Z-Axis:** Delay difference to idle

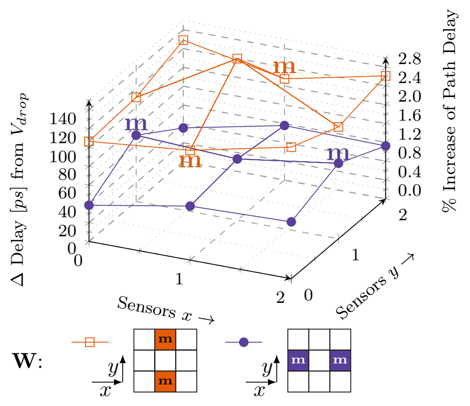
corners the worst-case voltage drop is reduced when one of the two workloads is phase-shifted. In Figure 3.16a (—□—) we can also see an increase in path delay from inactive to active regions of up to 4.1×, from 24.5ps in corner $x = 0, y = 0$ to 99.4ps in corner $x = 0, y = 2$, much higher than the 2.5× seen in the horizontal layout.

Additional effects that can be seen in the 3x3 layout is a difference in correlation on the vertical versus the horizontal axis. Figure 3.17 shows the results from two active regions on the edges of either the center row (—●—, $y = 1$) or center column (—□—, $x = 1$). The overall voltage drop is more prominent on the vertical axis, especially when no phase shift is applied. In that situation, a strong voltage drop amplification in the center region is visible, despite that region itself is not active. This gives valuable information for floorplanning and runtime scheduling.

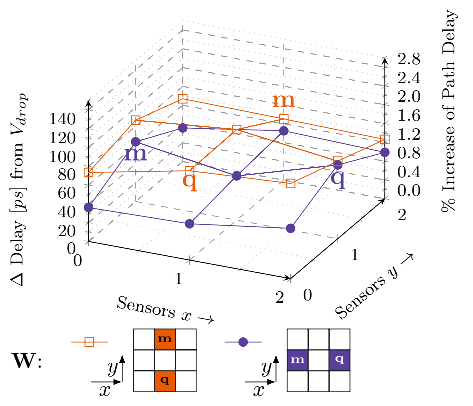
In Figure 3.18, we show two patterns of a fully active system (14% of all flip-flops). The patterns compare the difference between no phase shifting in one workload pattern (—□—) with phase-shifting the workload on every second region (—●—) by $\varphi = 50\%$. This phase-shifting can overall reduce the worst-case increase in path delay from about 5% to 2%. From these results it is also notable that in a fully active system, the center column (Regions with $x = 1$) is under the highest voltage drop condition. To conclude the results, we show two different patterns in Figure 3.19. The orange pattern (—□—) has less activity without phase shifts, but still shows a higher voltage drop than the more active purple pattern with phase shifts involved (—●—). Because the purple pattern adds a component of interference between some of the regions, these regions have less voltage drop over the others.

In this subsection we can thus confirm and extend some previous observations:

- Results in the previous sections showed that for our FPGA sample, activity in the



(a) Activity at the top and bottom of the center column (—□—) and left and right on the center row (—●—), without phase shift.



(b) Two center row (—●—) or column edge (—□—) regions being active, with one of them being phase-shifted by $\varphi = 50\%$.

Figure 3.17.: Average of multiple worst-case delay increases from voltage drop per sensor. These workload activity patterns show the effect of an activity in the edges of either the center column or center row, with one activity either 50% phase shifted or not. Workloads are summarized in Table 3.4. **X/Y-Axis:** Sensors across the FPGA. **Z-Axis:** Delay difference to idle

center region lead to higher voltage drop. We can confirm this result and extend it to a tendency of affecting the whole center column (x=1).

- Due to more distance across the FPGA, a difference in delay increase from inactive to active regions of up to 4.1× is shown, extending the previously seen 2.5×.

3.2.6. Discussion

The observations from our experiments allow us to formulate the following *design rules* or *recommendations*:

- During floorplanning, increasing distance among active regions as far as possible will decrease worst-case voltage drop.
- The activity in some regions has higher voltage drop impact compared to other regions. For instance in our tested Virtex-6 FPGA, an activity on center regions has higher impact on voltage drop, and should be avoided if possible.
- Distributing activity over time as uniform as possible will also decrease voltage drop.
- If multiple activities exist, spreading them both over time and space without overlap can be considered good practice.

These results provide useful insight to the design, how different workload behavior, as well as floorplanning, impact transient voltage drop and in turn circuit delay and the corresponding timing margin for correct operation. The results and analysis are a

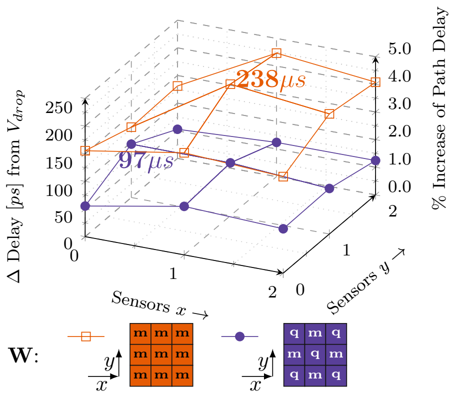


Figure 3.18.: Average of multiple worst-case delay increases from voltage drop per sensor. These workload activity patterns show the effect of all regions being active with (—●—) and without (—□—) a $\varphi = 50\%$ phase-shift for every second region. Workloads are summarized in Table 3.4. **X/Y-Axis:** Sensors across the FPGA. **Z-Axis:** Delay difference to idle

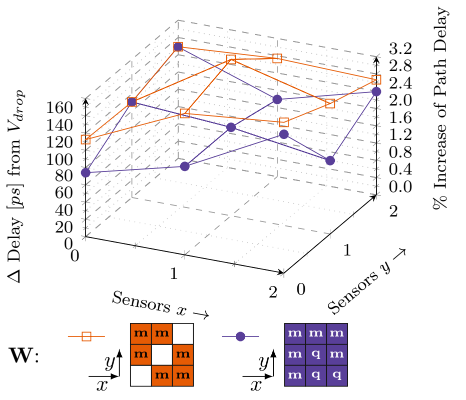


Figure 3.19.: Average of multiple worst-case delay increases from voltage drop per sensor. These workload activity patterns show the benefits of phase shifting with $\varphi = 50\%$ to reduce voltage drop, even if being applied asymmetrically (—●—) over less activity without phase shift (—□—). Workloads are summarized in Table 3.4. **X/Y-Axis:** Sensors across the FPGA. **Z-Axis:** Delay difference to idle

first step to guide design optimization in terms of both performance and reliability. In future work, we will focus on models to be used by designers to accurately guardband against such effects, and also be able to optimize their design to be resilient against them.

Our experimental results show a high dependency of voltage drop related to how activities are distributed in time and space, such that even overall lower activity through lower duty cycles or lower spatial distribution can induce more delay. This can be explained by:

- Less distribution in time means having more sudden increases in power and thus will lead to a higher di/dt voltage drop component, which dominates constant IR drops.
- Lower spatial distribution means higher power density and thus more load on the same number of decoupling capacitors.
- A switched-mode VRM typically operates at lower frequencies than the circuit, and therefore has an averaging nature. The shorter a certain timeframe of high power (i.e. switching activity) is, the higher voltage drop can be expected.

Having full control over the placement of active regions and type of running workloads enabled these observations. Real workloads would give a more convoluted view that is less clearly observable. In this way, we could see up to 3× difference in delay increase from two cases with the same overall activity (duty cycle), showing a high potential to mitigate this by scheduling. For example real-time systems typically work with fixed scheduling/workload periods, and their task execution times can resemble the used workload duty cycles.

The experiments involving spatial observations and interdependence reveal a spread of voltage drop across the FPGA. Furthermore, amplification and attenuation effects from positive and negative interference among distinct activities (i.e. workloads) were made visible. While voltage drop gradients have been analyzed in [50] for steady-state, we made the first analysis that includes transient voltage drops and the interdependence of a range of workloads. In combination with the first results, sufficient qualitative and quantitative evidence was collected to be useful for floorplanning and runtime scheduling decisions of on-chip activities.

Using the results and methodology from the first experiment, a system architect can find and work around the worst-case situation voltage drop hotspots (e.g. scheduling periods around $204.8\mu s$ in our case). On the other hand using the results from Experiment 2, a system designer can properly place and separate the active blocks, based on their expected switching activity, and therefore voltage gradient. These results can then be combined with Experiments 3-4, and lead to a joint strategy of scheduling block-level activity at design time and runtime. For a new system, our approach can be used to acquire similar results and use them for scheduling and partitioning decisions, or directly evaluate existing circuit designs for critical operating conditions. We believe that in a larger system, there is still sufficient freedom left for floor-planning on the coarse granularity of IP blocks to apply these strategies, while still satisfying all other constraints. The global interconnections between IP blocks are typically less than the connections within a block.

As the overhead of these sensors is fairly small, they can also be used during the system evaluation phase for monitoring the system behavior while running a mapped design with realistic workloads. This can greatly help if the FPGA is used in critical application domains with high reliability requirements.

On a final note, most of the overall observations and analysis are transferable to more recent technology nodes, ASICs, and processor designs, since they are based on similar underlying fabrication technologies, and share similar concepts in PDN design. PDNs do not scale or improve as fast as feature sizes do [17].

3.3. Chapter Conclusion

In recent technology nodes and complex designs, transient voltage fluctuations are a major threat to reliable circuit operation. Conservative safety margins and ad-hoc control mechanisms are too costly to handle this problem. A method to analyze the underlying effects inside the chip is required to gather more understanding of voltage transients.

This work showed such a method. By using sensors based on FPGA fabric, spatial and temporal voltage transients can be evaluated in-circuit. Our results show that an overall higher switching activity can lead to less critical voltage drop than a lower one, depending on the transient behavior in a full system that includes VRM. Considering the same activity, up to 3× difference in induced delay could be observed from transient voltage variance, while the spatial spread across our chip showed up to 4.1× delay variation from active to inactive region. Using our characterization approach, the acquired system-specific knowledge can be used as a guidance during circuit mapping as well as workload scheduling, leading to more reliable systems at increased performance. Future work will include the investigation of design implications of observed trends and how to optimize a design accordingly at physical design and runtime scheduling phases.

4. PDN-based Voltage Drop-based Fault Attacks

The work described in this chapter was published in [20], [21], and [25] and is joint work with co-authors (in no particular order) Fabian Oboril, Jonas Krautter, Falk Schellenberg, Amir Moradi and Mehdi B. Tahoori. More details on contributions is found in Section 1.1.

Fault attacks can break cryptographic implementations without the need of an algorithmic weakness, by injecting errors in the computation and gaining knowledge based on the faulty output. These attacks have been spread to various devices and types of algorithms [92–94]. A traditional requirement for these attacks has been unlimited physical access to the device. Before this thesis, the famous rowhammer attack has proven that software can also be used to cause faults. For that, the rowhammer attack uses specific memory access patterns, also affecting neighboring memory cells [95, 96].

In this chapter we will show that similar problems are more widespread and can also be applied inside FPGAs. The previous Chapter 3 has shown that specific switching activity can lead to higher or smaller voltage drops. Here we will show that such voltage drops can be escalated to cause actual faults in the system, both for injecting errors or bringing down the whole system. While in Chapter 3 we discussed only the reliability implications, we more specifically discuss it as a potential threat to security.

4.1. Voltage Drop-based Fault Attacks on FPGAs using Valid Bitstreams

In this section, we show a DoS threat that only requires partial access to the configuration of the FPGA, which can even be gained remotely. A vulnerability exists due to the possibility of generating excessive voltage fluctuations using valid bitstreams, and can cause a crash of the device within a short time. In the worst-case, the device will not reset itself, being a permanent DoS, until the FPGA or SoC is manually power cycled (power turned off and on). This can cause severe consequences, if the FPGA is used in servers.

So far, attacks on the electrical level have been carried out locally with access to the chip itself, for instance by manipulating its power supply or clock [66, 97]. Initial experiments showed that voltage drops can be generated from inside the system [44], but it has not been further evaluated. Older generations of FPGAs could be destroyed by overheating [67]. For modern FPGAs, overheat protection exists. The security vulnerability we expose in this section is more effective and malicious than simple overheating.

Our proposed attack uses a relatively small number of ROs to crash the FPGA (about 12% of total available Look-Up Tables (LUTs) used). Moreover, it just requires a few microseconds to crash the system, and thereby can escape various monitoring schemes. Additionally, it is not detectable by a thermal sensor, and finally, it keeps the FPGA inaccessible until it is fully power-cycled. This has been shown successfully in two different generations of FPGAs and could crash a full SoC, containing two processor cores with FPGA fabric, only using the vulnerability of this fabric.

The revealed vulnerability can be exploited for attacks with the goal to cause a DoS, affecting other users and applications. The focus of this section is to fully analyze this vulnerability and its overall impact. We analyze the exact conditions under which the system crashes, and how systematically it can be reproduced. Furthermore, we provide a first analysis and guideline for possible mitigation of this vulnerability.

4.1.1. Threat Model

The threat model we follow in this section considers an adversary with complete or partial access to reconfigurable fabric used in two different systems and usage scenarios. The adversary goals are to cause a denial of service in the system. One of the systems is a cloud-based environment, in which FPGA accelerators can be used by the adversary, another is a SoC with an embedded FPGA in which the programmable logic can be used (e.g. third party applications).

In a cloud-based environment, a DoS can make the FPGA unusable until manually power cycled, which is unlikely automated. In the case of a SoC with integrated FPGA, it has a similar effect, however if the system is battery-powered, the battery might need to be disconnected from the circuit. If it can not, the system stays in a permanent DoS.

The remaining part of this section is structured as follows: Section 4.1.2 explains the design of the required circuits for sensing and causing voltage drop and their experimental setup. Section 4.1.3 discusses the results and how these attacks could potentially be mitigated. Finally, we conclude the section in Section 4.1.4.

4.1.2. Circuit Design and Experimental Setup

In this subsection we explain the experimental setup of our components used in the analysis and the circuit we designed to expose the vulnerability. The implementation of voltage fluctuation sensors in our FPGA follow the concept introduced in Section 2.5. Here, we will explain how voltage drops can be generated through ROs in Section 4.1.2.1. Finally, we detail the rationale of the attack based on voltage emergencies and its experimental setup in Section 4.1.2.2.

In this work, we clock the sensors at $100MHz$ (i.e. one sample every $10ns$). The sensor values are transferred from our FPGA board to the PC using the Xilinx Chipscope Integrated Logic Analyzer. It saves the sampling data within FPGA BRAM and then transfers it to the PC.

We show the measurements from a sensor on the Virtex-6 FPGA, during a high voltage drop event in Figure 4.1. This figure shows a trace of sensor values (y-Axis) over time (x-Axis). Such a sensor value will show the *propagation depth* into the chain

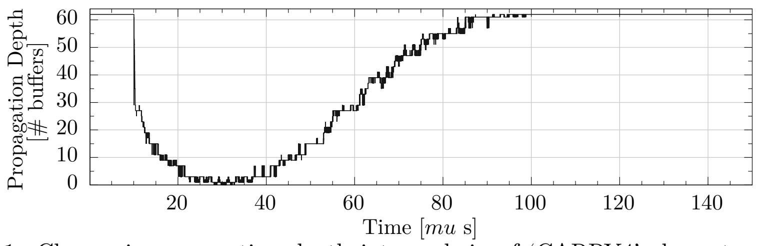


Figure 4.1.: Change in propagation depth into a chain of 'CARRY4' elements of a Xilinx Virtex 6 FPGA due to sudden current increase, caused by enabling 18720 ROs and keeping them enabled. Temperature 38–40 °C.

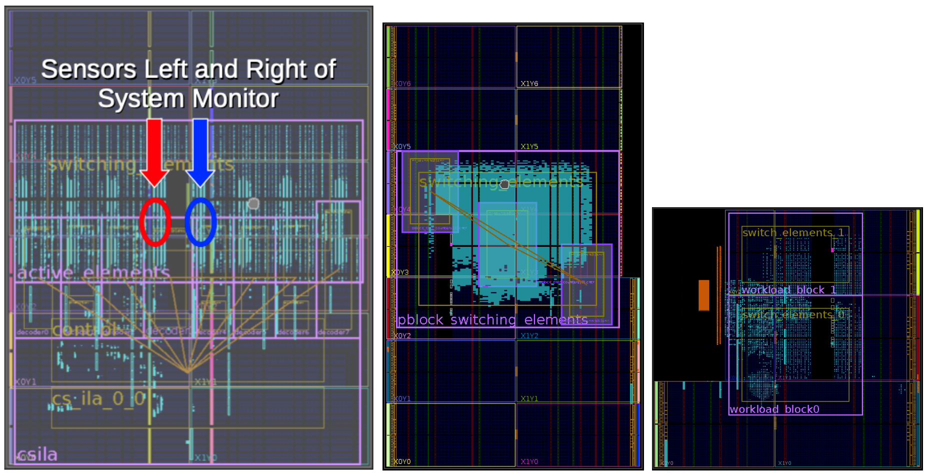


Figure 4.2.: Left to Right: Virtex 6 Floorplan with ROs and Sensors; Kintex 7 Floorplan with ROs; Zynq-7020 Floorplan with ROs.

of buffers, whose delay is affected by voltage fluctuations. Thus, higher values relate to faster buffers operating at a higher voltage, and as buffer delays increase from voltage drop, their propagation depth will decrease, shown as lower values on the y-Axis. Before the voltage drop within time $0 - 10\mu s$, the sensor is saturated at 62. At about $10\mu s$ the voltage starts to drop and the sensor shows that less bits can be propagated due to increased delay t_d of the transistors in the buffers. This voltage drop reaches its lowest point at $30\mu s$ and recovers back until $100\mu s$. It demonstrates how on-chip activity can lead to a strong increase in path delay. The delay is affected by the whole sensor range of the *Observable Delay Line*, corresponding to a change of about 12.5% of the total path (including the *Initial Delay*) according to timing analysis. In [44], 14% delay change was reported with a different method and FPGA. When sensors values are shown in the remaining work, they are based on two sensors, placed on the left or right side of the *System Monitor*, near the center of the FPGA, as shown in Figure 4.2.

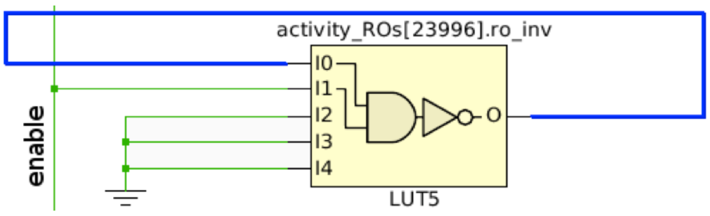


Figure 4.3.: Single ring oscillator made out of one LUT5 primitive

4.1.2.1. Generating Voltage Drops

We achieve the crash by causing timing faults or SRAM state retention loss through voltage emergencies. As explained in Chapter 2, Section 2.1, voltage drops are mainly dependent on changes in current and can therefore be caused deliberately when a spontaneous high current is required. In the setup explained here, high switching activity with proper timing is used to cause this excessive current and the resulting load on the PDN. LUTs are configured as inverters, and their input is connected to their output in a loop, forming ROs that can toggle very fast (i.e. as fast as physical transistor delays allow). By also configuring an AND in the LUT, an additional input 'I1' is used as an 'enable' signal to enable or disable the RO oscillation as shown in Figure 4.3. Standard FPGA tools can be used to implement these ROs.

By suddenly enabling all ROs through the LUTs 'I1'-inputs, they cause a strong influence on the voltage stability in the whole system, resulting in delay increases, especially in near-by paths. We showed this effect of voltage fluctuations on path delay in Figure 4.1, where within a few nanoseconds a steep drop can already be observed in a nearby sensor. Potential thermal influence is negligible in this observation, as with the low amount of ROs used in our experiments, temperature increases much slower and requires several seconds to minutes for a significant increase in heat. This means that such voltage drops cannot be detected by on-chip thermal sensors or slower integrated voltage monitoring circuitry.

4.1.2.2. Voltage Drop-based Fault Attack

Voltage drops depend on various other design and runtime parameters, such as process variation, local temperature and generic voltage noise. To cause a voltage emergency, an isolated voltage drop might be insufficient to create the required load on the PDN. Therefore we have to consider spatial spread and plan a series of various voltage drops carefully timed with respect to each other.

Consequently, we enable and disable the ROs to create voltage drop *pulses* in rapid succession, meaning they oscillate for a certain time and are deactivated in another, which we call the frequency of RO-toggling f_{RO-t}. This principle is depicted in Figure 4.4. Please note that this frequency of activation f_{RO-t} is different from the inherent switching frequencies of the ROs themselves. By varying f_{RO-t}, a reproducible crash of the FPGA can be caused. Please note too, besides this pulse frequencies, also the placement and amount of activated ROs has an influence on the attack quality. However, we observed that this influence is much lower than the pulse frequency, if a sufficient

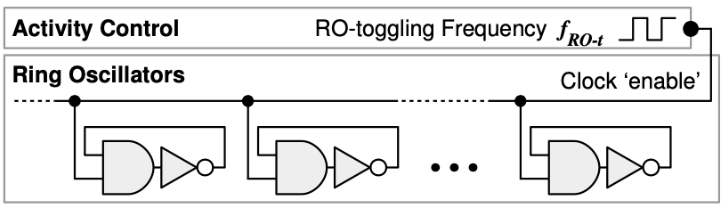


Figure 4.4.: Principle of connecting ROs to a clock f_{RO-t} that toggles their allowed activity (inside which they toggle themselves as fast as physically possible).

amount of ROs is available (when about >7% of the LUTs are used, tested on the ML605 board). Hence, in the following, we limited the description of our analysis to a fixed number of ROs, for the sake of brevity.

When testing different frequencies for f_{RO-t}, one needs to take into account that the worst critical situation takes some time to build up. When simply enabling the ROs, on the ML605 board, it takes about $10-20\mu s$ from the start of the path delay increase until it reaches the lowest point and starts to recover again, as shown in Figure 4.1. That means, we should keep the ROs oscillating until they cause a high enough voltage drop, and start over again, before voltage starts to recover. Due to that, reducing f_{RO-t} will crash the FPGA more effectively, until the range of recovery ($< 25-50$ kHz). To reduce the thermal influence on our experiments, we keep the FPGA within 38-40 °C using the on-board fan.

We also checked if timing failures could be caused when ROs are added to a legitimate design. Our investigation showed that it either operated correctly, or crashed entirely. Follow-up work is required to do a more detailed analysis.

4.1.3. Results and Discussion

The devices tested in this section are a 40nm Xilinx Virtex 6 on ML605 board (37,680 slices×4 LUTs), a 28nm Kintex 7 on KC705 board (50,950 slices×4 LUTs), and a 28nm Zynq 7020 on Zedboard (Dual ARM Cortex-A9 + 53,200 LUTs).

In the following subsections we will expose the previous mentioned vulnerability that can be exploited for DoS in these devices. We will first analyze the conditions required for crashing in Section 4.1.3.1, then we try to explain the root cause in Section 4.1.3.2. In Section 4.1.3.3 we then show if detecting a voltage drop with the delay line-based sensors is fast enough to prevent it, and discuss different possibilities to stop the activity that leads to the crash.

4.1.3.1. Crash Conditions

All three boards do not reliably crash when using a single voltage drop. But we can crash all three boards with an automated sweep through different f_{RO-t}, confirmed by a drop in measured power consumption on the board, and the stop of blinking LEDs controlled by the FPGA.

Of the three boards, the ML605 and KC705 always stop being accessible after the crash, and consume less power, which is due to a locked-up voltage regulator for at

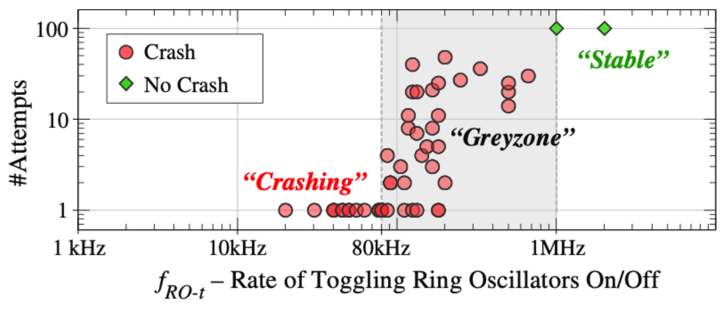


Figure 4.5.: Range of frequencies to toggle ROs on/off that cause the ML605 to crash after a certain number of attempts. Temperature regulated within 38-40 °C.

least the ML605. Thus, restarting the PC while keeping the board running does not resolve the situation. A manual power-cycle of the FPGA board is required before any access (i.e. JTAG) is possible again. On the KC705 we also tried to by-pass the on-board USB JTAG with a stand-alone JTAG dongle, which showed to be not sufficient to reactivate and access the FPGA board again.

The Zedboard has one of three conditions occur randomly. In one of them it stops being accessible to JTAG, like the other boards. In another case, it resets, which also deconfigures the FPGA part and resets the ARM cores. In the last case, it looks like a reset as well. However, when trying to reprogram the Zynq in Vivado, it locks up in the middle of reprogramming the bitstream. The software has then to be forcefully terminated. After that, the SoC stays inaccessible like in the first condition.

If the ML605 is connected to Chipscope during the crash, we get the following message:

```
WARNING: System Monitor Die Temperature has invalid data.  [..]  See Answer Record
24144.
```

Where a lookup of this *Answer Record* does not lead to relevant information. For the KC705 board, if connected to Vivado Hardware Manager, the software quits this program, and shows in a popup:

```
[Labtoolstcl 44-153] HW Target shutdown.  Closing Target:  localhost:[..]
```

For the Zedboard, when connected to the Vivado Hardware Manager and crashes in any way (inaccessible or not), the crash is not immediately recognized and will only be seen when trying to reprogram the board as:

```
ERROR: [Labtools 27-3165] End of startup status:  LOW
ERROR: [Common 17-39] 'program_hw_devices' failed due to earlier errors.
```

If it is connected for debugging during the crash, it will immediately show by:

```
ERROR: [Xicom 50-38] xicom:  Core access failed.  [..]
ERROR: [Labtools 27-3176] hw_server failed [..]
Resolution:  Check that the hw_server is [..]
```

If the ARM cores are on a debug connection in the Xilinx SDK, this connection is terminated.

In the following, we give more detailed results on the specific frequencies required for the crash on the ML605 board:

Table 4.1.: Different conditions for the tested boards. Power consumption measured with wall plug, default power supplies.

Board	% LUTs used for ROs	Standalone Power Consumption				Crash Recovery (Inaccessible)	
		After Reset	After Flashing/ Programming	All ROs Active	After Crash (Inaccessible)	PCIe Connected	Standalone
ML605	12.4%	14.3W	16.4W	36.5W	11.1W	Off/On Board	Off/On Board
KC705	11.8%	21.8W	13.5W	29.4W	7.0W	PC Power Off/On	Off/On Board
Zedboard	12.8%	3.3W	3.3W	6.2W	3.3W	not available	Off/On Board

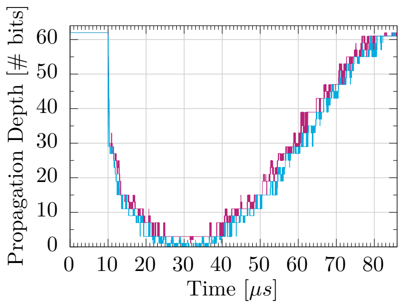
We activate 18720 ROs (about 12% of the LUTs in this specific Virtex 6) in a range of f_{RO-t} of 20 kHz−2 MHz. For each attempt to crash the FPGA, we program it and start an activity at a single f_{RO-t}. We then check how many attempts we require in order to crash the system. Since the recovery from a crash is time-consuming and not easily automated, we ran more experiments for the corner cases. The results in Figure 4.5 show how many attempts are required to cause a crash, depending on the chosen f_{RO-t}. Above 1 MHz, the voltage does not drop enough, and no crash happens with at least 99% probability (based on 100 attempts). Within 80 kHz−1 MHz is a 'greyzone' in which the crash occurs in a non-deterministic way. For the f_{RO-t} tested below 80 kHz, the crash happens always at first try.

To check the maximum time the crash requires, we set the Xilinx Chipscope Integrated Logic Analyzer (ILA) to trigger on the start of our malicious activity. After the trigger condition, we set it to collect only 16 samples before sending them to the PC. The expected time required in total from the trigger condition until received by the PC is less than 150μs, based on the JTAG frequency, data size and internal sampling rate. Thus, the crash happens in less than $150\mu s$.

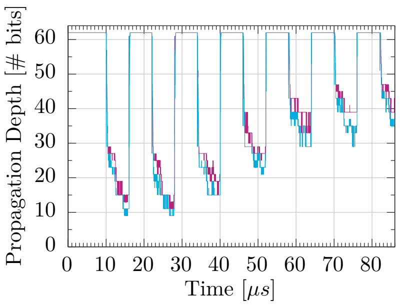
We additionally experimented with the two FPGA boards being connected inside a workstation to PCIe, in which the crash also happened. By using on-board switches to power either of the two boards off and on, the ML605 can be accessed again. However, the KC705 requires a cold reboot of the workstation (i.e. even the workstation's power supply), which can be a permanent DoS in a server environment that requires manual power cycling.

Interesting for these two boards is their power consumption after the crash, which is less than in any other condition by at least 20%, because one voltage regulator stopped operating. However, for the Zedboard there is no power difference. Table 4.1 summarizes the conditions.

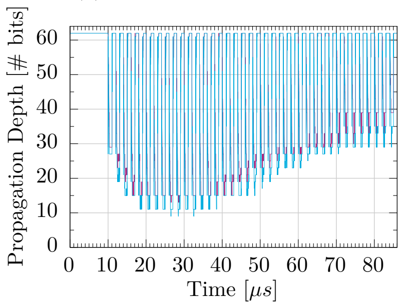
Exp.	Xilinx Virtex 6 ML605, 38-40°C, 18720 ROs	**Sensor**	—	Left of System Monitor
Setup:	based on LUT5, 100MHz Sensor sample rate.	**Position:**	—	Right of System Monitor



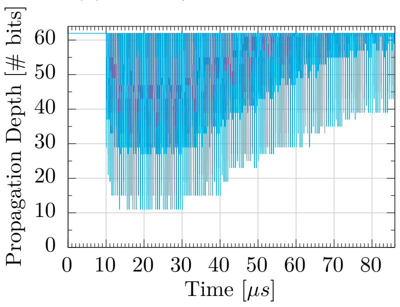
(a) Suddenly activate all ROs.



(b) ROs on/off at $83.333kHz$.



(c) ROs on/off at $500kHz$.



(d) ROs on/off at $2MHz$.

Figure 4.6.: Influence on the increase in path delay when applying different activation frequencies f_{RO-t} to the ring oscillators. Case (b) almost always leads to a crash, because at $80\mu s$ it is not yet recovered as (a) and the voltage drops last longer than (c) or (d). For (d) it never crashes, probably because the value never goes below 10.

4.1.3.2. Attack Analysis

The vulnerability exposed in this work is caused through voltage emergencies. In the background section, we reviewed various failure causes due to timing faults, SRAM state retention loss, or resonance in the PDN that supplies the FPGA. For the FPGAs tested here, both internal BRAM and the configuration memory are based on SRAM. The required time T_t and amplitude V_{drop} for a voltage emergency are different for a timing fault or SRAM state retention loss, where power distribution networks can have weaknesses when stressed at the right frequency.

On the ML605 board, $0V$ can be measured for the FPGA core supply voltage *VCCINT* after the crash. This situation shows that the respective on-board voltage regulator was shut off and is causing the permanent nature of the crash until power cycling of the board. The other voltage regulators on the board still operate normally and keep some LEDs lighted up and the fan spinning.

The sensors mapped to the FPGA can show some more details and evidence why this situation occurs, which we show with some example traces in Figure 4.6. These were collected when the system did not crash (therefore we can only show a minimum frequency of 83.333 kHz).

A larger V_{drop} can be seen when the ROs are activated suddenly and stay activated, as in Figure 4.6a, which recover after around $80\mu s$. When we cause repetitive events however, we can see a certain amplitude of V_{drop} to repeat itself for a longer stress period, shown for $f_{RO-t} = 83.333$ kHz in Figure 4.6b. When the frequency of f_{RO-t} is varied in Figure 4.6b-4.6d the worst-case V_{drop} of each frequency monitored with the sensors is very similar. However, the 500 kHz shown in Figure 4.6c does only rarely cause a crash, potentially because the sensors reach below a value of 10 only for very short times. With 2 MHz shown in Figure 4.6d we have never got a crash, and in that case, none of the sensors reach below 10.

Thus, a possible conclusion is that the board crashes directly due to extreme stress from a frequency-dependent resonance in the on-chip PDN or on-board voltage regulator. However, other components, like the *System Monitor* or configuration memory, might first become faulty, and subsequently lead to the voltage regulator getting disabled. The requirements to cause voltage emergencies in these components might differ from each other, and with that also the sensitivity to different f_{RO-t} frequencies. For instance, high frequency but short-timed V_{drop} 'spikes' can be absorbed in longer T_t times. Since in the experimented boards the memory and logic subsystems of the FPGA use the same supply voltage (and likely same PDN), the excessive voltage drop in the logic part (ROs) can cause voltage emergencies in the memory subsystem as well.

To collect evidence for a retention failure of SRAM, we can check either BRAM or configuration memory. We use BRAM, as it is easier accessible in the fabric, and Chipscope actually uses it for its sample memory. Thus we can see its failure by receiving corrupted data in Chipscope. We received such corrupted data, when experimenting with f_{RO-t} within the non-deterministic greyzone. In that range, we can sometimes still receive partial traces, short before a crash, showing evidence of an anomalous BRAM behavior.

In Figure 4.7 we show part of such a corrupted trace. In this trace until 131.66μs, we receive typical fluctuation data. After this time, all data bits received from Chipscope are '0' for exactly 150 ns, then they are '1' until the crash. In all the partial traces we

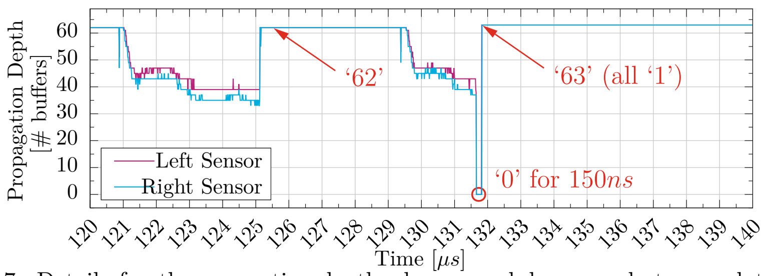


Figure 4.7.: Detail of path propagation depth when a crash happens, but some data was still transmitted. After the undershoot to all zero at $131.66\mu s$, all data received by chipscope is '1' – potentially the reset state of BRAM output latches.

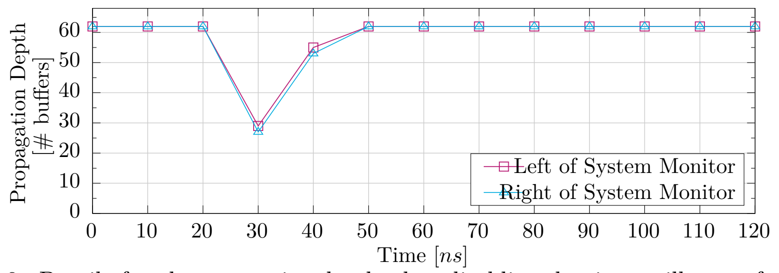


Figure 4.8.: Detail of path propagation depth when disabling the ring oscillators after detecting a drop in sensor value with previously idle activity.

recorded, the behavior is similar, and the intermittent ‘0’s always appear for 150 ns on a systematic basis.

Please note, in the case of receiving all-‘1’, it is not the sensor being saturated again, as the sensor saturation value is 62 and not 63, but the reserved part in BRAM is 6 bit. We assume itself, or its output latches, got reset to ‘111111’= 63.

In conclusion, the permanent nature of the crash depends on the voltage regulator crashing or shutting down for safety reasons. This situation is in turn either caused directly by resonance in the PDN, or by causing other problems in the logic and configuration memory of the FPGA. More details will be analyzed in future work.

4.1.3.3. Discussion

In the previous sections, we showed the nature of deliberately caused voltage emergencies that lead to a DoS. Specifically, the attack leads to a DoS situation much quicker than by overheating, and with the addition of keeping the FPGA inaccessible, even to JTAG, until its power supply is reset.

Allowing user-configurable accelerators in such systems create security vulnerabilities that can compromise the availability of FPGA resources. A complete server might require reboot, including full power cycling. In the worst-case, a system based on SoCs with included FPGA can get into a permanent DoS, for instance when they are powered by a non-removable battery. Threats like these could be reduced, given a scheme to detect and disable the excessive switching activity, before it escalates to voltage emergencies for the complete chip.

In an additional experiment, we show how fast voltage recovers when all ROs are disabled after a sensor value below ‘30’ is detected. This way, we estimate the latency of the sensor and how fast the voltage drop wears off. Figure 4.8 shows the sensor readings for this experiment. The sensor values show recovery after two samples ($20ns$). However, as the sensors are saturated at 62, the system might still require more time to fully recover.

Applying this option to a real design would require to reserve area for the sensors and one input on each LUT. The sensor thresholds would need to be chosen such that legitimate activity is not affected. Additionally, all LUTs would effectively have one input less available to the design, and no option for a disable switch might exist for other FPGA primitives, making it rather infeasible and requiring other options These options could be based on power gating or a way to quickly disable the interconnects in affected areas.

To even prevent malicious bitstreams from loading, they can be sanity checked in software, with the challenge of keeping legitimate bitstreams working, but not leaving loopholes for malicious ones. One option that recent FPGA tools already use as default constraints is checking for combinational loops during bitstream generation, which can be deactivated by the user. Thus, the check would need to be done at a privileged system software level, inaccessible to the user or application developer.

For all of these possibilities, new experiments are required. To be able to deactivate arbitrary malicious circuits fast enough, new FPGAs might need to be manufactured.

4.1.4. Section Conclusion

FPGAs become more widely adopted as user-defined accelerators, such as in the cloud, or integrated in SoCs. In these new usage scenarios, classic ways to enforce security, like bitstream encryption, become infeasible, making them vulnerable to new security threats. In this section, we revealed such a security vulnerability, by showing a systematic approach to crash two generations of FPGAs, and an SoC containing an FPGA, which can lead to a denial of service threat in these systems. This denial of service is caused by a specific configuration when bitstream-level access is given, and only requires about 12% of the available LUT-resources in the tested FPGAs. Such a vulnerability can allow attacks on FPGAs used in data centers, SoCs, and other application domains, where the entire system needs to reboot or even be fully disconnected from power in order to power cycle the crashed FPGAs. Additionally, we discuss how proper mitigation could be implemented to prevent denial of service.

4.2. FPGAhammer: remote voltage fault attacks on shared FPGAs, suitable for DFA on AES

In this section, we present *FPGAhammer*. In analogy to rowhammer, we cause faults through repetitive activation patterns. Similarly to Section 4.1 we affect the supply voltage of an FPGA. In Section 4.1 it was shown to cause voltage drops to crash an FPGA-system to perform DoS, while this section shows how precise timing faults can be injected. FPGAhammer is precise enough to inject timing faults in FPGA logic, suitable to target specific encryption rounds of the AES and perform DFA. We carry out and elaborate this attack on various FPGA boards, containing Intel Cyclone V SoCs with different configurable logic sizes. We conclude that it is indeed possible to induce timing faults in a cryptographic core through remote configuration, with a partial bitstream that can be easily generated with official FPGA vendor tools.

In summary, this section makes the following contributions:

- We introduce a new category of software-initiated fault attacks in FPGA systems, possible with remote access to the target only, based on supply voltage drops generated by means of malicious yet legitimate switching activity.

- We establish a generic threat model for an attacker and a victim using a shared FPGA resource in an active fault attack scenario.
- We show that a spatially and logically separated attacker in one region of the FPGA fabric can attack a victim in another region.
- We test and prove the general vulnerability to on-chip voltage drop fault injection on a range of FPGA platforms, and elaborate an automated way to inject faults more precisely.
- We empirically prove that fault injections achieve a precision high enough for a successful DFA and key recovery on the AES, regardless of FPGA model or process variation within the tested devices.

The remaining section is structured as follows: Section 4.2.1 explains the proposed threat model, the related work, and background on how voltage fluctuations occur inside chips. Moreover, we briefly outline the DFA method we apply in our attacks. In Section 4.2.2, we present an initial attacker design and describe the behaviour of FPGA boards from different manufacturers under the influence of malicious switching activity. In Section 4.2.3, we elaborate on how a fault attack and subsequent DFA can be carried out with the proposed method on an FPGA AES implementation. An overview of the hardware used in the experiments and implementation details are provided. We also present results of analyzing injection rates and key recovery success. We discuss some ideas for future research based on our findings in Section 4.2.4 and conclude our work in Section 4.2.5.

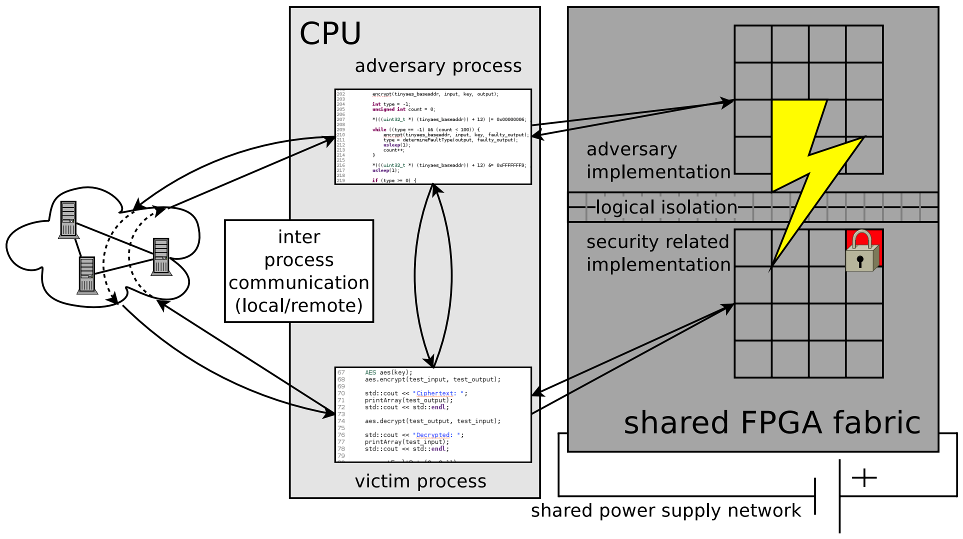
4.2.1. Preliminaries

Before elaborating a full attack on the AES, we put our work in the context of other publications, which are relevant to our findings or assume a similar threat model. Moreover, we briefly explain the theoretical background of our experiments and the basics of causing a voltage drop, leading to faults in FPGAs.

4.2.1.1. Threat Model

Here, we further describe the attacker-victim scenario assumed throughout this section. A brief overview on this scenario is given in Figure 4.9. We assume the victim and the adversary to have access to a fraction of an FPGA, in which they can load their own arbitrary design, like a cryptographic accelerator. Both attacker and victim have their respective processes in an operating system, and their designs on the FPGA are logically and spatially separated, and follow other common best practices as explained in [98]. It is also assumed that the respective FPGA fabric is powered by a single common power supply. This scenario includes both data-center applications, in which FPGAs are utilized as standalone accelerators, as well as SoC platforms, in which multiple processes on the CPU can utilize a fraction of the FPGA logic.

We assume the victim to utilize their part of the programmable logic for a security related algorithm, such as a block cipher. A secret key used in this algorithm is either hard-coded onto the FPGA or transferred at runtime.


CPU
adversary process
inter process communication (local/remote)
victim process
adversary implementation
logical isolation
security related implementation
shared FPGA fabric
shared power supply network
+

```
        encrypt(tinyaes_baseaddr, input, key, output);

        int type = -1;
        unsigned int count = 0;

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) |= 0x00000006;

        while ((type == -1) && (count < 100)) {
            encrypt(tinyaes_baseaddr, input, key, faulty_output);
            type = determineFaultType(output, faulty_output);
            usleep(1);
            count++;
        }

        *(((uint32_t *) (tinyaes_baseaddr)) + 12) &= 0xFFFFFFF9;
        usleep(1);

        if (type >= 0) {
```

```
    AES aes(key);
    aes.encrypt(test_input, test_output);

    std::cout << "Ciphertext: ";
    printArray(test_output);
    std::cout << std::endl;

    aes.decrypt(test_output, test_input);

    std::cout << "Decrypted: ";
    printArray(test_input);
    std::cout << std::endl;
```

Figure 4.9.: Overview of the threat model considered in this work: Attacker and victim share an FPGA resource with a common power supply network, but isolated, logically disconnected partitions on the fabric

If we consider a symmetric encryption module, such as an AES implementation on the FPGA, used by the victim, we assume the following **Adaptive-Chosen-Plaintext-Scenario**:

- The adversary can issue arbitrary plaintexts to a public interface of the victim process either locally or remotely through a network.
- The victim outputs the ciphertext of the provided plaintext, encrypted with the secret key, only known to the victim.
- The amount of requests is limited only by the attackers computational restrictions, which we assume to be polynomial.

We remark that we assume an attacker that can encrypt arbitrary plaintexts in the generic threat model. However, the actual content of the plaintext is irrelevant for a successful DFA and may even be unknown. The attacker only needs to be able to enforce encryption of the same plaintext twice, where one case is a fault-free encryption and in the other case faults are injected. Therefore, a DFA based attack on AES like the one presented in this section, can be applied to situations where replaying encryption requests to the target module is possible. Please keep in mind that attacks with faulty ciphertext only are also possible [99], but are subject to future studies to reveal whether they are feasible by internal voltage-drop based fault generation in FPGAs.

4.2.1.2. Fault Injection using FPGA logic

A functional block in a synchronous FPGA design includes a common clock signal, which is used to synchronize all memory components within the block. This means that combinational paths between registers (D-flipflops) are constrained in their delay by the clock signal. If a signal takes longer than a clock cycle to traverse a combinational block, timing violations may cause the output of the target register to be different from the desired results – a timing fault occurs.

The constraints can be formulated with five parameters [66]: The clock cycle time t_{clk}, the internal register delay d_{clk2q}, the setup time t_{setup}, which is the amount of time an input signal has to be stable at a register input, the maximum data propagation time through the combinational logic d_{pMax}, and the clock skew t_{skew}, which is the phase difference of the clock signal between two different registers.

The timing constraints can then be expressed by Equation 4.1.

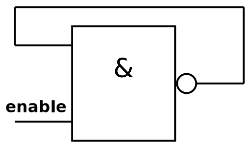
$$t_{\text{clk}} > d_{\text{clk2q}} + d_{\text{pMax}} + t_{\text{setup}} - t_{\text{skew}} \tag{4.1}$$

A higher data propagation time can be achieved by lowering the power supply voltage V_{DD} [28]. The increased delay raises the right hand side of the above equation, leading to a timing violation and a potential fault injection.

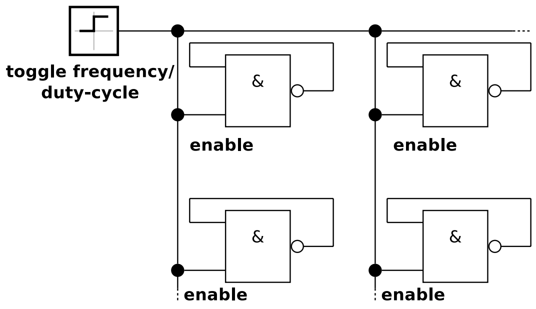
To understand how the supply voltage of an FPGA can be decreased with on-chip logic elements, it is necessary to understand how the PDN of an FPGA behaves under the influence of different designs on the fabric. The PDN includes a network from the voltage regulation module on the board down to the internal power rails and every transistor on the FPGA. Generally, the PDN can be modelled as a mesh of resistive, inductive and capacitive elements. Therefore, the power supply voltage depends on two parameters: The average static current drawn by the implemented design (IR-drop) and the voltage drop caused by switching activity and inductance (di/dt-drop). The relation is summarized in the *Law of Inductance*: $V_{\text{drop}} = IR + Ldi/dt$. With technology scaling, the effect of static voltage drops has become less relevant compared to the voltage fluctuations caused by switching activity and inductive components [28].

To evoke malicious switching activity on an FPGA and cause an excessive di/dt-drop, we can deploy a massive amount of ROs to generate high frequency current oscillations. In Section 4.1, ROs were already used to induce voltage drops high enough to crash FPGA-based systems. It was shown, how a singular activation of a large amount of ROs causes the voltage to drop rapidly by a certain amount and then more slowly return to the original value within about 50 μs for the tested devices. Moreover, it was described, how a drop can be increased significantly, by driving the entire grid of oscillators with a different, slower frequency, constantly enabling and disabling the ROs. A dependency on the duty-cycle of this RO toggle signal was demonstrated as well in Chapter 3.

ROs are implemented by connecting any odd number of inverters circularly. An additional enable signal allows enabling and disabling the oscillation, to connect each RO to a common toggle signal. Figure 4.10a shows the schematic of an RO with an enable signal and a single two-input NAND gate. The frequency of the oscillation depends on the gate delay and the loopback routing from the output of the gate to



(a) A single RO, implemented with a NAND gate



(b) The implemented oscillator grid, used to cause voltage drops

Figure 4.10.: Schematics of the RO implementations

the input. Since the gate is usually implemented as a single LUT in the FPGA, the oscillation frequency depends mostly on the loopback routing.

4.2.2. Provoking Faults in FPGA Designs

Before developing a full DFA attack on the AES, we reproduce results from Section 4.1 about provoking crashes on Intel FPGAs, which has only been shown on Xilinx devices so far. Moreover, we evaluate fault injection vulnerability of a broad range of FPGA devices and boards from different manufacturers.

4.2.2.1. Initial Design for Causing Voltage Drops

For stressing the PDN and inducing a voltage drop, we deploy an RO grid onto the FPGA fabric, which can be enabled and disabled with a variable toggle signal. A schematic overview on the design with multiple ROs is presented in Figure 4.10b. The characteristics of the voltage fluctuation caused by the oscillators depend on the toggle signal frequency and duty-cycle.

Figure 4.11 shows a trace of the externally-measured supply voltage of an Intel Cyclone V SoC device, when performing a sweep over different toggle frequencies for the RO activation signal, until the device crashes. We toggle the RO grid with a decreasing frequency, and for each frequency, increase the duty-cycle up to 75% until the voltage fluctuation significantly exceeds the supply voltage limits of the device. After this point it crashes, leading to a hard reset of the included ARM Hard Processor System and a loss of the configuration of the FPGA device.

When developing in Hardware Description Languages (HDLs), the synthesis software from most FPGA manufacturers deems the RO design entirely useless. Thus, it is necessary to prevent the synthesis tools from optimizing the RO grid away. Initially, we conducted experiments with different oscillator designs and found significant differences regarding their effectiveness in causing the supply voltage to drop. We studied the following design options on Intel FPGAs:

a) Implementation using an output pin: The first approach is to connect all

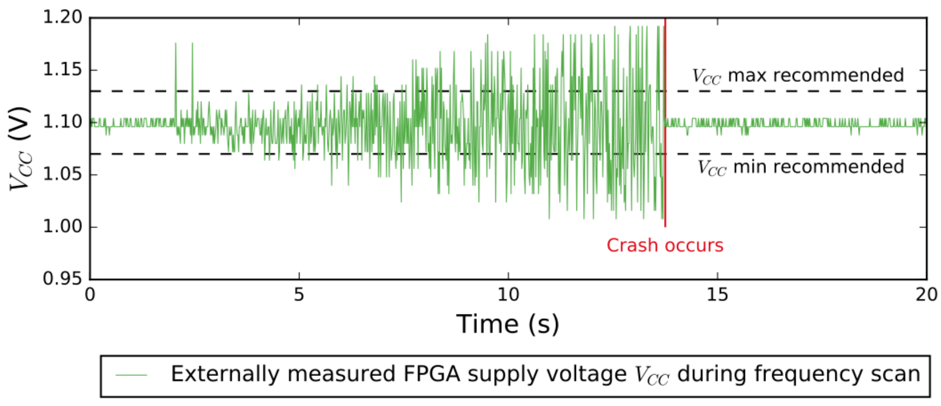


Figure 4.11.: Trace of FPGA supply voltage *VCCINT*, measured externally with an oscilloscope during a frequency sweep leading to a crash of the device

ROs through a reduction function to an arbitrary output pin of the FPGA. This approach was discarded immediately, since the routing congestions limit the amount of ROs significantly.

b) **Implementation using virtual pins:** Secondly, we can declare all outputs of the ROs as virtual pins, which are implemented as a single LUT each on the FPGA.

c) **Implementation without additional elements (Bare ROs):** Another possibility is to define output connections of each RO to the top-level entity, but not to an output pin. The synthesis software then accepts the fanout-free LUTs without additional elements.

In Figure 4.12, we present results of comparing the two RO variants with (b) and without (c) virtual output pins, where each implementation has the same amount of logic utilization. We stressed a simple test design, which is detailed in the next section. We collect the number of faults in a series of 10 trials for 5 seconds each. The amount of recorded errors in the test design proves the higher effectiveness of the virtual pin

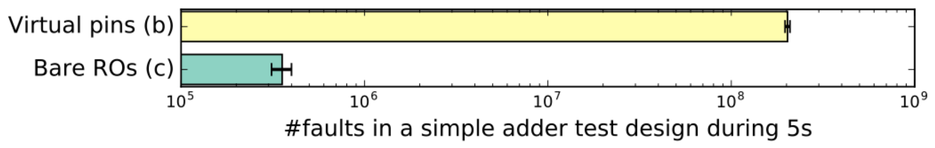


Figure 4.12.: Amount of errors detected in a simple adder test design during 5 seconds of RO toggle activation with respect to the RO implementation option. Tested on DE1-SoC.

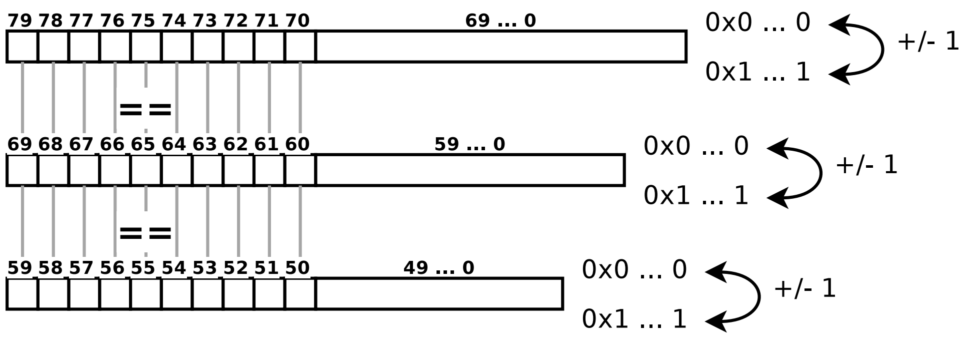


Figure 4.13.: Simple adder design to evaluate fault attack vulnerability

option. Despite less ring oscillators are used in variant b, the additional interconnect resources connected to each single oscillator cause more faults.

Voltage drops can be further increased by enforcing high interconnect utilization when separating ROs from their respective virtual output pins. However, we have observed that the design which works the best, strongly depends on the used device, especially if devices from different manufacturers are considered.

4.2.2.2. Voltage Drop-based Timing Faults in a Simple Test Design

Before applying fault attack analysis to a cryptographic implementation on an FPGA, we analyze the behaviour of various FPGA boards, while stressing them with a massive amount of ROs. We find that all evaluated boards from different manufacturers are susceptible to the oscillator grids in terms of showing unusual behaviour upon activation of the oscillation.

An overview of a simple adder design for the evaluation of fault attack vulnerability is depicted in Figure 4.13. Three register carry-chains of different lengths are driven in a way that keeps them switching between their maximum and minimum values. When the maximum value is incremented, the resulting overflow requires a carry bit to be propagated from the LSB to the MSB through the entire carry-chain. Likewise, decrementing the minimum value causes propagation of a carry bit through the registers.

In the given design, we can now compare n of the uppermost bits of each adder with the correct and expected result. When a voltage drop increases the propagation delay of circuit elements, this comparison shows whether a timing fault occurred in the respective adder.

We then investigate the behaviour of the adder design with different lengths and under various frequencies. In these configurations, it is possible to find a setting, where the timing analysis of the FPGA mapping tools shows a violation of the timing constraints, yet the design works in a normal situation. ROs can then cause timing faults at runtime.

In production-stage cryptographic implementations, users are likely keen to avoid timing-violations reported by the analysis tools during design synthesis, which makes the vulnerability of devices in this situations less relevant to an adversary. We found

Vendor	Board	Device	Fault attack possible (constraints unmet)	Fault attack possible (constraints met)
Intel	Terasic DE4	Stratix IV	Yes	Yes
Xilinx	XUP PYNQ-Z1	Zynq-7000	Yes	No
Lattice	iCE40HX8K-B-EVN	iCE40HX8K	Yes	Yes
Intel	Terasic DE1-SoC	Cyclone V SoC	Yes	Yes
Intel	Terasic DE0-Nano-SoC	Cyclone V SoC	Yes	Yes

Table 4.2.: First results about general vulnerability of different platforms

that on most of the tested platforms it is also possible to inject faults into designs that meet the timing constraints of worst-case estimation models.

In Table 4.2, we summarize our results across several platforms, which we evaluated regarding the feasibility of fault attacks in designs that do not meet the timing constraints (unmet) and designs that meet the constraints. Although some platforms seem to be not vulnerable, it is more likely that we simply failed to find the appropriate parameters to activate the oscillators in a way to trigger the necessary voltage drops yet. These initial results promise a possible success regarding the application of fault injection and DFA to cryptographic modules on FPGAs.

4.2.3. Fault Attack Evaluation on AES

In this subsection, we illustrate the details of performing a full key recovery attack on AES using the RO design for fault injection and the DFA explained in Section 2.9 for key recovery. We prove the concept of on-chip fault attacks by evaluating fault injection and key recovery on the Intel Cyclone V SoC chip family. The following subsections describe the general setup and detailed parameters for each experiment and present the acquired results.

We initially detail how we achieve the required fault injection precision with an automated calibration approach. Subsequently, we investigate the general fault injection rate with respect to the amount of ROs in the attacker design and inter-die process variations of three DE1-SoC boards. We continue by evaluating the success rate of a full AES key recovery for 5000 keys. Additionally, we study the dependence of injection rates on the operational frequency of the AES module on the smaller Cyclone V SoC device on the DE0-Nano-SoC board.

4.2.3.1. Calibrating the Fault Injection Precision

The DFA attack from [75], which was described in Section 2.9, allows to recover a secret AES key with only two pairs of correct and faulty ciphertexts, if the fault is injected according to the fault model of a single byte fault before the 8th round of the AES encryption. In practice, we need to adapt parameters such as frequency, duty-cycle,

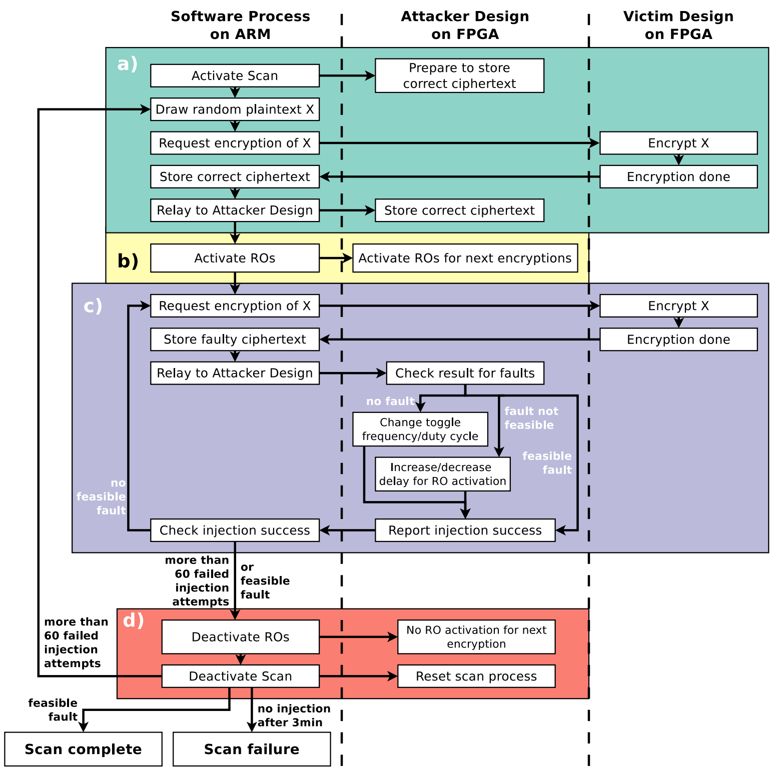


Figure 4.14.: Flowchart of the calibration algorithm to find the appropriate parameters for injecting faults at the desired moment

and activation time of the RO grid to provoke faults at the proper moment of the encryption.

Any fault before the 8th round of the AES leads to all bytes of the faulty output ciphertext to be different from the correct one, whereas any fault after the 9th round leads to less than four bytes to be different. Faults that are injected into the input state matrix of the 9th round are revealed in exactly four bytes of the faulty output ciphertext being different from the correct one. This allows us to verify a successful fault injection using the output ciphertext. Therefore, we decide to aim for injecting faults not before the 8th round of the AES but before the 9th round only.

To make use of the possibility for injection success verification, we develop an automated calibration algorithm, to be executed before evaluating injection rates or attack success for a given design and device. The algorithm allows to use the attacker design in different setups, without the need for finding appropriate parameters in time-consuming trial-and-error experiments manually.

In Figure 4.14, we present an overview of the full calibration algorithm we use before evaluating injection rates or key recovery success. We adapt the signal for activating the ROs in three parameters: The toggle frequency, the duty-cycle and the delay between

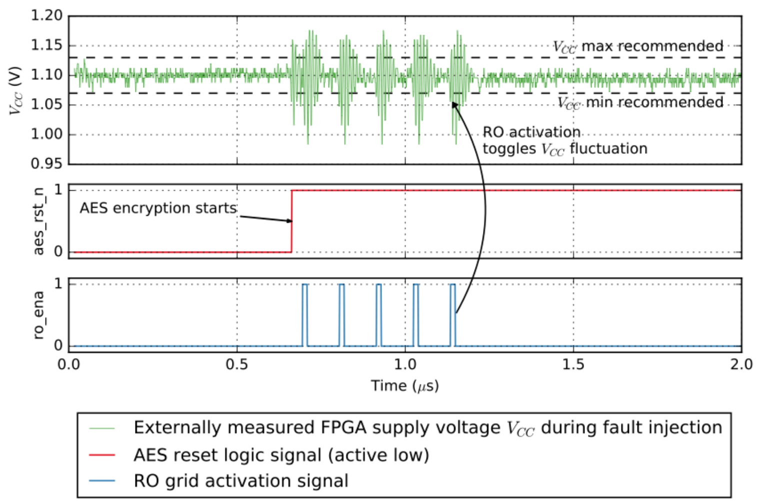


Figure 4.15.: Trace of FPGA supply voltage *VCCINT*, measured externally with an oscilloscope during a single fault injection attempt

starting the encryption and activating the RO grid. On the left side of the flowchart, we depict the process flow on the software side, whereas the right side enlists the actions carried out on the FPGA by both attacker and victim design. The algorithm performs as follows:

a) The attacker activates the calibration process on the FPGA. A random input plaintext is drawn and encrypted without RO activity. The result is stored as the correct ciphertext.

b) Afterwards the fault injection process on the FPGA is activated, to toggle RO activity for the following encryptions.

c) Encryption of the same random plaintext is requested. The attacker design on the FPGA activates the RO grid with an initial frequency and duty-cycle and no activation delay. If no fault is detected, the frequency/duty-cycle are decreased/increased. Duty-cycle is increased for each frequency up to 75%. If a fault is detected, which affects an undesired subset of bytes, the injection occurred too early or too late, and the activation delay is increased/decreased. In any case, the attacker design reports the injection success to the attacker process, which either requests another encryption or continues the process.

d) If the injection was successful or a predefined maximum of injection attempts inj_{max} were unsuccessful, the attacker software deactivates the RO grid and either finishes the successful scan or chooses another random plaintext for fault injection.

During experiments, we determined $inj_{max} = 60$ as an appropriate upper limit regarding the number of encryptions needed until a fault is injected, since not all random plaintexts result in a faulty output even with the ROs active. After a successful calibration process, the three parameters are fixed and used for subsequent fault injections.

In Figure 4.15, we show an externally acquired trace of the FPGA supply voltage VCCINT during a single fault injection by the attacker design on the FPGA. The AES reset signal (aes_rst_n), which resets the AES encryption module when low, indicates the start of an encryption. To provoke a fault, the attacker design pulses the RO grid (ro_ena signal) with the previously determined frequency, duty-cycle and activation delay. The voltage fluctuations (V_{CC}) cause a critical delay at the desired moment with a higher probability and therefore, a fault is injected.

In conclusion, our approach requires eight instead of only two ciphertexts compared to [75] to recover the secret key, but makes the attack feasible in practice, where a lot more encryption requests are required to have the attacker design affect the AES module at the desired moment. In all subsequent experiments, we apply this variant of the attack, aiming to inject faults before the 9th encryption round. The calibration is executed only once, at the beginning of any evaluation. However, we continue to filter faults that have been injected at an earlier or later encryption round during the collection of ciphertext pairs. This method maximizes key recovery success, although the injection precision achieved by our calibration is very high, as we show in the results.

Furthermore, we remark that the acquired calibration parameters can even be reused on different devices of the same type. Therefore, there is no need to perform specific calibrations on the board on which the fault attack is to be performed. In our experiments on the Terasic DE1-SoC for example, we find that a toggle frequency of 1.16 MHz with a 56% duty-cycle is selected most frequently in a run of 1000 calibrations. Fixing those parameters and reusing the design on a different DE1-SoC board leads to similar or even better fault injection rates, depending on the general vulnerability (process variation) of the board.

Note that this method filters out all faults that are caused at the wrong AES encryption round but does not necessarily discard ciphertexts, which result from fault injections that affect multiple bytes before the 9th round. Some multi-byte faults can also lead to four faulty bytes at the desired positions in the output ciphertext. Therefore, we still have some amount of keys, which can not be recovered during evaluation, because multi-byte faults are not covered by the theoretical fault model of the DFA. Further details regarding unsuccessful key recoveries can be found in the results in Section 4.2.3.4.

4.2.3.2. Hardware and Software Environment

In this subsection, we specify the devices, which have been used in the further experiments, and provide details on the implemented hardware designs and software tools.

The AES implementation we use as a proof-of-concept in this work is a simple, small module for 128 bit key length encryption. It utilizes around $300 - 400$ registers and about $750 - 850$ LUTs in the tested Cyclone V FPGAs and takes 50 clock cycles to encrypt a given plaintext. The module is not protected against side-channel or fault attacks.

We perform our experiments on systems based on the Intel Cyclone V SoC family,

Silicon speed (Process variation)	Operation temperature
Fast	0° C
	85° C
Slow	0° C
	85° C

Table 4.3.: STA corner timing-models available in the Quartus STA from fastest to slowest model for a given device speed grade at 1100 mV supply voltage

which incorporate an Intel FPGA and a 925 MHz Dual-Core ARM Cortex-A9 processor inside a single die. The 5CSEMA5F31C6 chip is embedded on the Terasic DE1-SoC board. We studied injection rates and attacks on three Terasic DE1-SoC boards of different age and usage history to account for process and aging variation. A smaller variant of the Cyclone V SoC, the 5CSEMA4U23C6N, with only half the amount of logic elements is present in the Terasic DE0-Nano-SoC board, which we investigated as well. The devices are used with their standard, unmodified power supplies.

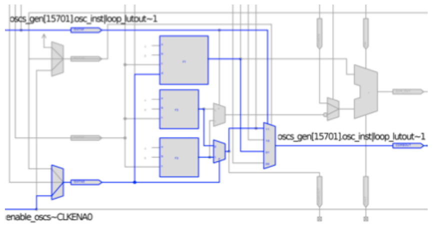
Both boards have an SD card slot, which we use to boot a Linux system and run user applications, that interact with the FPGA fabric, on the ARM processor. We encapsulate the AES cryptomodule as an Intel Avalon Memory-Mapped slave device, which allows access from programs running within the Linux system on the CPU.

The Intel Quartus Prime software offers tools for Static Timing Analysis (STA), which analyzes the design in terms of timing violations under four different models (corners) for a given device with a specific speed grade [100]. We enlist the available timing-models in Table 4.3. The fast/slow classification of silicon for the given device speed grade refers to propagation delay variations caused by intra-die process variation.

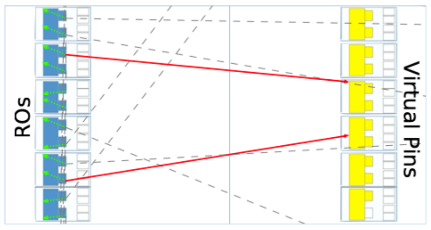
If the timing analysis reports timing violations at the time of implementation, the design is not guaranteed to work reliable under all operation corner cases in terms of temperature and voltage levels, according to official chip specifications as found in the FPGA datasheet. We focus on attacking designs that do not violate any timing constraints, even at the worst-case 85° C corner, but investigate the influence of worst case path slack in designs that violate the constraints as well. For each experiment, we explicitly report whether timing constraints are violated in the respective subsection later.

We implement the attacker design as a grid of ROs as described in Subsection 4.2.1.2. A single RO is composed only of the combinational part of a single ALM on the Cyclone V. The output is directly routed through local interconnect back to the input. This way, we achieve the fastest possible switching frequency for the oscillators. In Figure 4.16a, an example of how the Intel Quartus Prime software synthesizes and fits a single RO into the bottom part of one ALM can be examined. The used output on the right and input on the top left of the ALM are the same and the additional enable signal is connected to the bottom left input of the ALM.

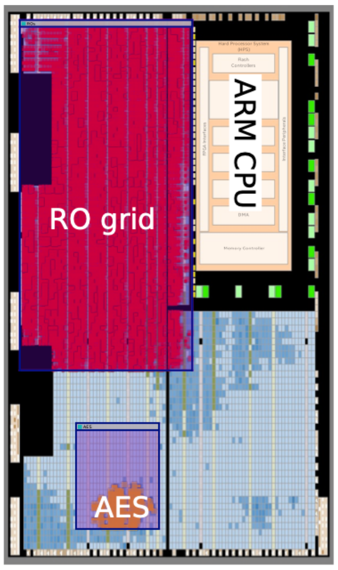
The Intel Quartus Prime software reports the worst case delay through the LUT to be about 0.08 ns and the loopback routing delay through local interconnect around


oscs_gen[15701].osc_inst|loop_lutout~1
oscs_gen[15701].osc_inst|loop_lutout~1
enable_oscs~CLKENA0

(a) An RO as implemented during compilation in the LUTs of an Adaptive Logic Module (ALM) of the Intel Cyclone V SoC



(b) ROs in a Logic Array Block (LAB) of the Cyclone V SoC on the left with the interconnect to their respective virtual output pins on the adjacent LAB on the right (continuous lines) and other LABs (dashed lines) as well as loopback routing (dotted lines)



(c) Design for evaluation of fault injection after fitting as displayed in the Quartus Chip Planner on the Terasic DE1-SoC board with the AES module in the bottom area and the ROs grid in the top left region

Figure 4.16.: Implementation details of the RO attack design on the Intel Cyclone V SoC as displayed in the Quartus Chip Planner

0.21 ns. Therefore, assuming a maximum delay of 0.3 ns through gate and loopback, the RO can achieve frequencies of 3 GHz and more.

As explained in Subsection 4.2.2.1, an RO-based design with a virtual pin (variant b) is most efficient to provoke critical voltage drops, which is why we choose this design variant for our attack on AES as well. In Figure 4.16b, we show how several ROs defined as an oscillating LUT and a virtual output pin are mapped into a LAB of the Cyclone V SoC as presented in the Quartus Chip Planner. The schematic shows the loopback routing (dotted lines) of each RO and the routing to their respective virtual output pins, two of which are placed in an adjacent LAB on the right (continuous lines) and some in different regions of the FPGA fabric (dashed lines). The relevant Verilog code parts for implementing an RO grid on Intel FPGA devices can be found in the Appendices Section A.1, Section A.2 and Section A.3.

Moreover, it is necessary to drive the oscillator grid with a very specific frequency and duty-cycle. We therefore add an enable signal to trigger each of the implemented ROs, which is routed through a global clock buffer on the Cyclone V SoC. The use of this type of signal, originally intended for distribution of clock signals on the FPGA, allows to save on routing resources from the toggle frequency control design block to all of the ROs and accelerates the compilation of the entire design significantly.

Since our attack scenario, elaborated in Section 4.2.1.1, assumes a shared multi-user FPGA use-case, we constrain each design using the LogicLock feature of the Intel Quartus Prime software to keep victim and attacker design blocks within designated areas of the FPGA fabric. To avoid any variation from other components on the chip, we additionally activate the region reservation parameter of the LogicLock region, that contains the AES module, which prevents the fitter from placing any other logic than the AES module and its Avalon MM encapsulation into this area. Figure 4.16c shows the ROs mapped into the top left area and the AES module in the bottom region as displayed in the Quartus Chip Planner for the design on the larger Cyclone V SoC on the Terasic DE1-SoC. On the software side, we implement tools for controlling encryption and fault injection to be executed within the Linux system on the ARM core of the Cyclone V SoC. The evaluation of the collected ciphertext pairs and respective DFA is performed on a standard host computer with an Intel i7-7700HQ Quad-Core processor.

4.2.3.3. General Fault Injection Efficiency on the DE1-SoC

In order to evaluate the general fault injection efficiency, we first generate bitstreams for different percentages of logic utilization of the attacker logic in the range of 30% to 50% for the DE1-SoC board. The victim module runs at a frequency of 111 MHz, which does not violate any timing constraints as explained in the previous subsection, even in the worst-case corner of the static timing analysis. Then we measure the number of faults occurring for one million encryption requests from a previously generated set of random plaintexts, which are reused for all experiments. We evaluate the experiment on three different Terasic DE1-SoC boards of different age and usage history. The encryption key remains the same for one test series, which is repeated for a second random key for each of the DE1-SoC boards. Before every evaluation, the calibration algorithm as described in Section 4.2.3.1 is executed, to find optimal parameters for provoking the desired kind of faults, which can be used in the subsequent DFA.

Figure 4.17 shows the total number of faults out of 1M trials F_{tot}, as well as the number of faults usable for DFA with our described fault model F_{DFA}. Both results are shown dependent on the number of activated ROs in % of available LUTs in the FPGA. We see that both F_{tot} and F_{DFA} initially increase at the same rate, starting from a different minimum amount of required ROs for each board respectively. This proves the effectiveness of the calibration algorithm before each evaluation. On all boards we see, how the calibration algorithm is able to adapt to a variety of different setups with a very high precision. However, if the amount of activated ROs exceeds a certain level, the effect is too strong to allow precise injections. F_{tot} still increases with more ROs, but can affect more than one round or byte per round. Hence, the resulting ciphertext will have more than four byte faults, which can not be used anymore to recover the secret AES key in the used fault model.

4.2.3.4. Total Key Recovery Success Rates on the DE1-SoC

Subsequently, we evaluate the success of the full DFA attack including recovery of the secret AES key. This evaluation reflects on the success of our entire algorithmic flow

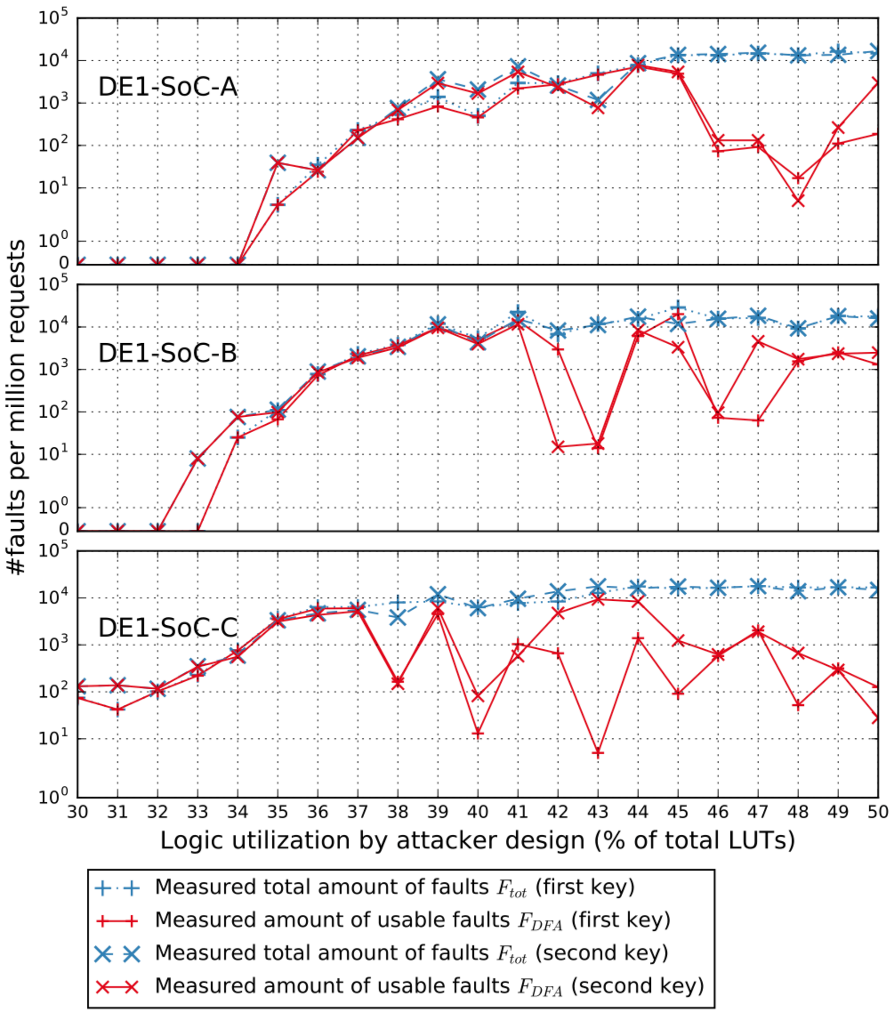


Figure 4.17.: Total measured fault injection rates F_{tot} and measured injection rate of faults usable in DFA F_{DFA} with respect to the amount of logic utilization (percentage of total LUTs) by the attacker design for three different Terasic DE1-SoC boards and two different random encryption keys

of injecting faults before the 9th AES encryption round, the calibration algorithm and filtering of undesired faulty ciphertexts. For each of the three boards, which we already used to investigate fault injection rates, we use the amount of ROs that lead to the highest injection rate F_{DFA} of faults usable for DFA. Again, the AES module has an operating frequency of 111 MHz, where no timing violations are reported by the STA. We generate a set of 5000 random AES keys and collect a minimum of two ciphertext pairs, which exhibit faults at the desired positions, for each four bytes of the last AES

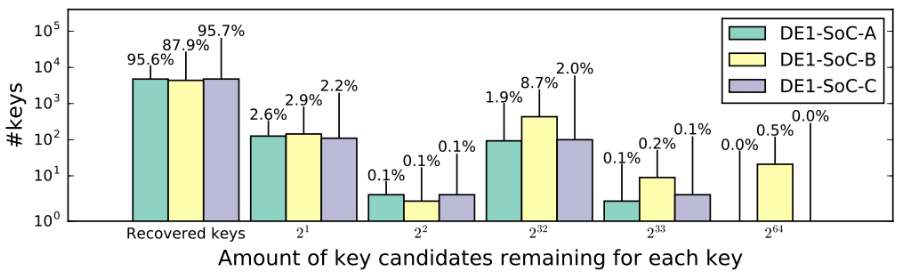


Figure 4.18.: Amount of key candidates remaining for each DFA key recovery attempt on 5000 randomly drawn AES keys on three different DE1-SoC boards

round key. The ciphertexts along with each key are stored on the SD card of the board and later transferred to a host computer. After collecting the minimum amount of faults required, we apply the DFA from [75] with the slight adaption of assuming single byte faults before the 9th round instead of the 8th round. In that case, we require a minimum theoretical limit of 8 ciphertext pairs per key.

Figure 4.18 summarizes the results of the key recovery attempts on our three DE1-SoC boards. On all three boards, we are able to deploy the attack completely successful for at least 87.9% of the 5000 random keys. All recovered keys are correctly recovered, so no false positives are encountered. On all boards, we have a small amount of around $2-3\%$ of all keys which can not be recovered, but less than four candidates for the last round key remain. This ratio confirms the results in [75], showing that in about 2% of the cases more than two ciphertext pairs are necessary to recover the AES key. If a sufficiently small amount of key candidates remains, the correct key can be easily recovered with an exhaustive search. We encounter, however, some keys, where more than 2^{32} or even 2^{64} candidates remain. Across all our experiments, an average of 22 usable faults were required to gather the required two ciphertext pairs per four bytes of the round key. To collect these pairs, the attacker design needs to issue 17979 encryption requests on average to the AES module, which took on average 2344 ms. The average time for the evaluation of one attack until key recovery on the described host machine is about 107 ms.

Ultimately, the attack can therefore recover a secret AES key in about 90% of cases. In the remaining cases, fault injection itself fails. Our calibration algorithm and subsequent filtering of faults, which can not be used in DFA, prevents the gathering of faults that have been injected at any other stage of the AES encryption than before the 9th encryption round. However, as mentioned in Section 4.2.3.1, the method is unable to distinguish some multi-byte from single-byte fault injections. The adapted fault model from [75] assumes single byte faults before the 9th encryption round, which is why key recovery attempts are unsuccessful, if the faulty ciphertext is the result of a multi-byte fault.

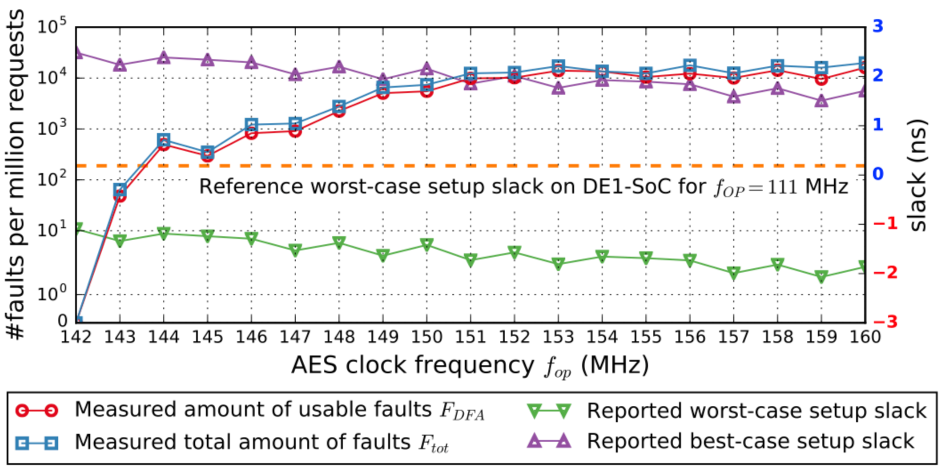


Figure 4.19.: Total measured fault injection rates F_{tot} and measured injection rate of faults usable in DFA F_{DFA} as well as setup slacks reported by Quartus STA for different AES operating clock frequencies f_{op} with preserved design placement but remaining routing randomization on the DE0-Nano-SoC

4.2.3.5. Slack-dependent Fault Injection Vulnerability of the DE0-Nano-SoC

The DE0-Nano-SoC provides about half the amount of logic elements than the DE1-SoC. Since the power supply is, as the experiments suggest, equal or at least similar to the one of the DE1-SoC, we need to utilize a huge percentage of LUTs to attack a design, which does not violate any timing constraints in the four timing models during the Quartus STA. Therefore, we additionally study the fault injection rates using an attacker design which always occupies only exactly 50% of logic resources with respect to different operational frequencies f_{op} for the victim AES module, assuming an FPGA split exactly between two users. The STA reports violation of the two worst-case timing corners (slow silicon speed, 0°/85° operating temperature), whereas the timing constraints are fulfilled within the fast silicon speed models.

We implement the evaluation design as described first with an initial clock frequency of $f_{\text{initial}} = 160$ MHz for the AES module and its ARM interconnect. We direct the fitter to use maximum effort to fulfill the given timing constraints for the AES module, therefore optimizing placement of logic elements w.r.t. timing constraints. Then we prevent the fitter algorithm from changing the placement of logical components within the partitions for each recompilation of the design. The randomization during the placement algorithm can make a design running on a higher frequency cause less timing violations, than a design running on a lower frequency. However, since the routing can not be fixed completely, the routing algorithm still causes some derivation from the desired outcome.

The operating clock frequency f_{op} for the mapped and placed design with remaining routing variations is decreased in steps of 1 MHz to a frequency of $f_{\text{op}} = 142$ MHz and for each design the fault injection rates for one million total encryption requests are recorded. Furthermore, we note the setup slack values for each design as provided by the STA in the worst-case corner (slowest silicon, 85° C) and in the best-case corner (fastest silicon, 0° C). For reference, we also show the worst-case path slack value for the design running at 111 MHz on the DE1-SoC.

In Figure 4.19, we show the fault injection results together with the respective slack values at different operating frequencies. The results show that the reported worst-case and best-case slacks for the design do not directly correspond to the respective operating frequency f_{op} linearly, due to the remaining heuristic algorithm in the routing stage. However, the trend is that slack values increase for lower frequencies and decrease for higher frequencies. Both F_{tot} and F_{DFA} increase together with operating frequency f_{op}. However, the increase is more steep within the threshold range 145 MHz $\leq f_{\mathrm{op}} \leq$ 151 MHz. A divergence between F_{DFA} and F_{tot} with increasing frequency is not as significant as in the experiments with respect to logic utilization. Single experiments with f_{op} = 170 MHz imply, however, that the injection becomes less precise with increasing f_{op} as well. We were unable to inject faults into the design running at f_{op} = 142 MHz.

4.2.4. Discussion and Future Work

These results show the effectiveness of provoking voltage drops with ROs and launching a DFA attack. The most effective RO design makes use of virtual pins and benefits from added load through toggling activity on interconnect wires. However, in this section we just elaborated three variants of RO designs to cause high load. As the results suggest, future work should look into toggling more interconnect wires. Alternatively, we may consider other on-chip methods for provoking a decrease in supply voltage. For example in [101], short circuits are caused by maliciously crafted bitstreams. Those illegal bitstreams can, however, be detected by software tools easily. Furthermore, the calibration algorithm can be replaced by more advanced methods, possibly including machine learning approaches, to make injection as precise as possible. On the other hand, a more generic fault model, that covers all possible occurring faults before the 9th AES encryption round, may also raise the key recovery success rate to 100% [102]. Using self-heating elements for example from [103], we may also increase the temperature of the FPGA remotely, improving the fault injection rates as shown for AVR microcontrollers in [104].

Just like in [105], where a passive side-channel attack on the integrated ARM processor of a Xilinx Zynq SoC was carried out through voltage monitors on the FPGA, we also need to consider extensions of our given threat model to hardware that is tightly coupled with the FPGA, and shares a significant part of the power supply network. In Section 4.1, crashes of a full SoC could also be provoked from the FPGA part of the system. From that, we can conclude that these systems are basically vulnerable to RO based attacks. In the future, it will have to be proven if fault attacks are also possible in this configuration, or if there are reasons that the integrated CPU will crash before showing any timing violations.

Because these attacks are a risk to any multi-user FPGA applications, proper countermeasures will have to be investigated in the future. Fortunately, a remote attacker that has only access to the FPGA fabric is more limited than an attacker with full physical access. Thus, significantly increasing timing margins might be a sufficient countermeasure to this attack, with the possibly high cost of reducing the speed of the circuit. However, adding arbitrary timing margins just reduces the risk, but does not ensure no timing violations. To give more guarantees, delay elements can be added to an FPGA design, that will invalidate an output result when the design is close to timing faults [106, 107]. In these cases, the cost of reducing the circuit speed might also be less, with only small area overhead.

Internal sensors allow the user to detect critical path timing failures in their designs, which has been used for dynamic voltage scaling features and detecting transistor aging [45, 108]. Similarly, sensing timing failures can enable detection of voltage drops and delay line sensors have been already shown to detect possible fault injection attempts for local fault attacks [44].

On the FPGA hardware design level, in the generic threat model of remote attacks on FPGAs, vendors might consider to only allow shared FPGAs when the regions of each user are residing in their own respective voltage island. However, this might defeat the purpose of an efficient way to utilize FPGA resources.

Another idea, that previous works already mentioned briefly, would require to check each bitstream that is loaded to an FPGA, as briefly discussed in Section 4.1 and [105]. By essentially restricting the available basic logic circuits and, for example, prohibit or limit the implementation of combinational loops and ROs, the attack presented in our work could be mitigated. In a more complex attacker design, however, an algorithm to verify the bitstream for potentially malicious implementations would not be limited to polynomial complexity, since attacker logic can also be hidden within legitimate logic, as existing research on *hardware trojans* suggests [63]. Furthermore, the reduced flexibility of FPGA implementations may be too big of a disadvantage for this restriction to be considered as a countermeasure.

Some of the mentioned ideas might already be sufficient for mitigation, but finally, we also believe that our work is just the beginning, and more effective attacks are yet to come.

4.2.5. Section Conclusion

FPGAs are getting increasingly adopted in larger computing systems, as accelerators in the cloud or SoCs. In such scenarios, multiple isolated users will share a single FPGA fabric and PDN. Previous works have already shown power analysis side-channel attacks in this scenario, without requiring dedicated sensors. In this section, we additionally prove fault attacks on shared FPGAs possible by applying a DFA attack on the AES in a similar scenario, also implemented with standard FPGA tools. We demonstrated that FPGAs will not just crash if voltage drops are injected with ROs, but in fact timing faults can be injected with sufficient precision for DFA. First, we showed an effective way to inject timing faults in simple test designs with the focus on reducing required FPGA resource use. Based on this method and given precision of the injections, we adapted an existing fault model for AES, and performed a successful DFA with a fully automated calibration of the injections. We evaluated the general injection rate and precision with respect to the percentage of logic utilization by the attacker design and the operating frequency of the target AES module. Evaluating the injection rates showed how an attacker can provoke sufficient faults for a key recovery, with logic utilization in the range of only 35% to 45% of the LUTs on a Terasic DE1-SoC board based on an Intel Cyclone V SoC. Since our calibration algorithm allows precise injections, independent of target parameters, we were able to recover at least 90% of secret AES keys from a set of 5000 randomly drawn keys on three different boards. The results in this work highlight the importance of further research, before FPGAs can be adopted widely in multi-user scenarios.

4.3. Further Fault Attack Investigations

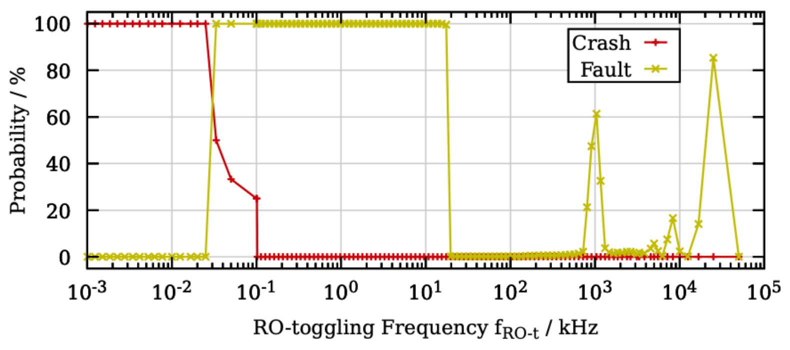
In an extended analysis carried out for [25], other devices were tested and characterized to find out which conditions for DoS or fault injection are necessary.

4.3.1. Fault Attacks on Xilinx Ultrascale VCU108

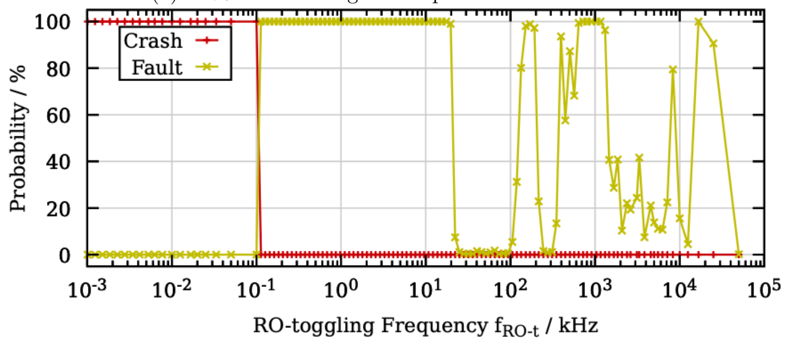
For most tested Xilinx FPGA Boards, DoS is possible using ROs by causing a very strong voltage drop, with 8 %–14 % of the FPGA LUTs configured as ROs in the Virtex-6 ML605, Kintex-7 KC705 and Zedboard Zynq-ZC7020 platforms. This DoS attack does not just reset the circuit programmed into the FPGA, but completely clears its configuration memory such that reconfiguration is necessary to restore its function. Using a specific coding style, ROs can be implemented without any intervention from the Xilinx Vivado software [109].

For the VCU108 Virtex Ultrascale board we perform more detailed experiments to characterize the precise probabilities per frequency to cause a crash or induce timing faults in another part of the FPGA. The timing faults are evaluated by implementing a long 288-bit adder on the FPGA, which has a remaining slack of 0.037 ns. During operation it alternates between a large number addition and subtraction that cause overflow or underflow to stimulate the critical path. With as low as 25 % of FPGA LUTs used for implementing ROs, timing faults can be reliably caused with $f_{RO-t} = 9.6 .. 13.7\,\text{kHz}$. At 20 % LUT usage we only get a reasonable number of faults with $f_{RO-t} = 25\,\text{MHz}$ with 87 % probability. 15 % LUT usage could not cause faults on our tested board.

When increasing the LUT usage to 30%, both crashes and timing faults can be caused, depending on the ROs-toggling frequency f_{RO-t} as introduced in Section 4.1, Figure 4.4.We show more detailed results in Figure 4.20, that shows the respective probabilities for a timing fault or a crash. For each frequency, we launch a burst of 512 clock periods of f_{RO-t}. To find the fault probability, we check the result of our 288-bit adder for correctness and take a normalized average. For the crash probability, we launch 512 of these bursts and plot the percentage how many of them caused a crash. Here, the FPGA is reliably crashed with an f_{RO-t} at 200 Hz or lower frequencies, while about 200 Hz to 10 kHz reliably induce timing faults. We also show the results when the ROs are increased from 30% to 35% LUT usage in Figure 4.20, in which a more clear boundary between fault and crashes can be seen. Anyhow, one can observe a wide frequency band for which it is safe to perform timing faults without the risk of crashes that would require reprogramming the bitstream. Thus, fault attacks to compromise



(a) **30%** of LUT usage for implementation of ROs.



(b) **35%** of LUT usage for implementation of ROs.

Figure 4.20.: Range of frequencies to toggle ROs on/off that cause timing faults or crashes in the Xilinx VCU108 Virtex Ultrascale Board.

system integrity become feasible without accidentally causing a crash that would be more obvious to the victim.

4.3.2. Fault Attacks on Intel Terasic DE1-SoC

In addition to the VCU108, we conducted experiments on an Terasic DE1-SoC Evaluation Board with Intel Cyclone V.

We employ 23 % of FPGA LUTs for ROs to cause voltage drops in the same way as on Xilinx FPGAs. The ROs are enabled at frequencies $f_{RO-t} = 1 \ldots 4000\,\text{kHz}$. Depending on these frequencies, the DE1-SoC either crashes or timing faults are observed in an AES core implemented on the device. We launch multiple frequency sweeps with bursts of 512 clock periods of f_{RO-t} per frequency. Above 80 kHz, the FPGA is operating stable, while below 80 kHz, the FPGA crashes with high probability. Below 30 kHz we observed crashes in all our experiments. In a separate experiment, $f_{RO-t} = 1..3\,\text{MHz}$ causes timing faults in an AES module.

5. PDN-based Power Side-Channel Analysis Attacks

The work described in this chapter was published in [22], [23], [24] and [25] and is joint work with co-authors (in no particular order): Falk Schellenberg, Jonas Krautter, Amir Moradi and Mehdi B. Tahoori. The works in [22], and [23] have also been included in the PhD Thesis of Falk Schellenberg, in [110]. More details on contributions is found in Section 1.1.

In this chapter we present three main works looking into the PDN as an attack vector that can be exploited remotely through software access. By that, the security of a full system can be undermined. We show this problem in three main stages:

- Section 5.1 shows that power analysis side-channel attacks are feasible within an FPGA chip.
- Section 5.2 shows that these attacks can even be escalated to a full board-level system, such that one FPGA can attack another Integrated Circuit (IC) on the board that shares the same power supply.
- Section 5.3 shows a software-based power analysis attack within a mixed-signal chip, and by that generalizes shared power supplies as a new type of attack vector within an IC.

5.1. An Inside Job: Remote Power Analysis Attacks on FPGAs

Cryptographic devices often deal with secret information as well as privacy of the users. So-called Side-Channel Analysis (SCA) attacks target the implementation of cryptographic schemes and are independent of their mathematical security. For example, [111] exploits the response time of an RSA implementation to retrieve the used secret key. Introduction of Differential Power Analysis (DPA) attacks [112] resulted in extensive research in refining attacks and developing countermeasures. Although timing attacks might even work over the Internet, power analysis attacks are thought to require physical access to the device, i.e., to connect an oscilloscope to measure the power consumption or the electromagnetic emanation in the near proximity. Yet, in the following, we prove this assumption to be wrong. This falls well within the line what has been seen for fault attacks. Before Rowhammer [95], fault attacks were thought to require some sort of physical access to induce a fault into the target. Instead, the attack can lead to pure-software based privilege escalation from an underprivileged user.

Furthermore, it can be introduced remotely as well, even at a very high abstraction level [96].

For side-channel attacks, the dynamic power consumption originating from the switching of transistors is usually targeted. Our methodology is based on the work presented in [18], in which a mechanism to capture the fluctuation of the internal supply voltage of FPGAs is shown. It is in fact shown that the supply voltage at different locations of a PDN is not constant and depends on the activity of the logic. We followed the same principle and built internal sensors to locally monitor the dynamic change in the supply voltage. As a proof of concept, we conducted our experiments on a Spartan-6 FPGA, where an AES encryption module as the targeted cryptographic core is implemented. Indeed, the results show that such sensors enable side-channel attacks retrieving the secret key and is nearly as powerful as when using an external measurement.

We highlight two important properties of the proposed attack enabled by these sensors: a) it does not require a signal connection to the targeted core and b) it can be implemented using general-purpose logic available on any FPGA. These properties lead to a large threat when using 3rd party IP cores, considering both ASIC and FPGA implementations. Furthermore, it enables power attacks in emerging use-cases for FPGAs, such as fabric being shared among multiple users in the cloud [4] or the FPGA being part of a complex SoC. In such scenarios, an attacker might be able to deploy a voltage sensor unnoticed, essentially acting as a hardware Trojan to spy on the power consumption of the unaware victim. This seems contradictory to one of the motivations of using FPGA fabric as accelerator for cryptographic primitives in SoC: although enabling SoCs to receive security patches in hardware well after the design cycle [113], it may open doors to the previously unknown threats and attacks.

Considering the related works, the first powerful hardware Trojan has been presented in [114], where the Trojan inserted at HDL level of a CPU design would give the attacker unlimited access to the CPU resources. Further examples include malicious designs (at netlist and HDL level) made public during the student hardware Trojan challenge ICCD 2011 [115] or stealthy Trojans at the layout level [116]. Other works like [117–119] have shown methods to build malicious designs which only leak out the secrets when the attacker conducts specific SCA attacks. In [120], a design methodology for building stealthy parametric hardware Trojans and its application to bug attacks has been proposed. Other works, including [121, 122], propose Trojans which are triggered by aging or reduced supply voltage. In [116], a Trojan is embedded into an SCA-resistant design, and would result in the cryptographic keys leaking through the same side channel but only under a particular condition, e.g., by means of a certain power model.

The Trojan we present in this work aims at leaking the cryptographic keys through a side channel as well. However, what makes our work different to the state of the art is its remotely accessible feature. In short, we present how to design power consumption sensors – synthesizable with FPGA primitives – which can be placed in another module next to, or even far from, the cryptographic core.

The remaining section is structured as follows: In Section 5.1.1, we elaborate the adversary model in more detail and explain the required background knowledge regarding PDNs and the voltage sensors. Subsequently, in Section 5.1.2, we explain the

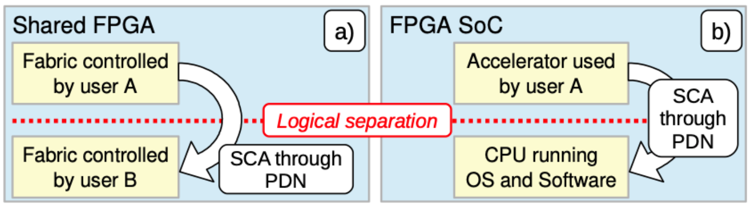

Shared FPGA
a)
Fabric controlled by user A
Logical separation
Fabric controlled by user B
SCA through PDN
FPGA SoC
b)
Accelerator used by user A
SCA through PDN
CPU running OS and Software

Figure 5.1.: Two scenarios of SCA attacks, where the circuits are logically separated, but share the same PDN. **a)** In a shared FPGA, one user (A) can attack another (B). **b)** In an FPGA SoC, a user with current access to the FPGA accelerator can attack any software or operating system on the CPU.

implementation of the sensor and provide a discussion of the experimental results in Section 5.1.3, while Section 5.1.5 gives a section conclusion.

5.1.1. Preliminaries

5.1.1.1. Adversary Model and Threat Analysis

Considering Figure 5.1, we assume two scenarios for the adversary in this section. In both systems, the adversary's goal is to extract secret information from the other system components, with only access to the PDN, and no signal connections. In the first one, the adversary has partial access to the FPGA fabric, whose resources are shared among multiple users, e.g., FPGA accelerators shared in the cloud [4]. When the sensors are hidden in a complex application – which for instance needs to communicate with the outside of the FPGA – the inspection over the design would not detect any connection between the cryptographic module and the rest of the application, i.e., low chance for the Trojan to be detected. Automated isolation and verification countermeasures for FPGAs with user-controlled logic especially in data centers have already been proposed in [123]. Such techniques usually employ some physical gap between different IP cores with well-formed interfaces [98, 124]. Yet, we later show that such a barrier might be breached with internal voltage sensors, even when the sensors are placed far away from the target.

In the second scenario, an attacker has full access to the FPGA while the FPGA is part of a large system like an SoC where CPUs reside on the same die. For example, we can recall reconfigurable fabric of an SoC, where 3rd party users are allowed to use the FPGA. Any underprivileged user with access to the FPGA fabric can embed the sensors, thereby potentially monitoring the voltage of the whole SoC. This is an increasing threat under the trend of accelerator-use.

Note that, as opposed to the previous works about covert channels passing through this isolation, e.g., by electrical coupling [125] or even by temperature [126], we do not alter the attacked IP core in any form and only monitor the unintentional power consumption. The effects of electrical coupling have been further investigated in [127, 128].

5.1.1.2. Voltage Drop Sensors

While many FPGAs feature dedicated internal power sensors, they are shown to be inappropriate for side-channel analysis [129]. Regarding the realization of suitable custom sensors based on reconfigurable logic, Zick et al. [44] showed an implementation that uses existing FPGA fabric to sense variation in supply voltage, based on the concept of measuring the propagation time of a signal with TDCs, as explained previously in Section 2.5.

In this section we follow the same concept. The idea is to use a delay line, in which a clock signal propagates through a chain of buffers. As the delay of these buffers depends on the supply voltage, the buffers can be monitored as a surrogate of it. The delay line can be *tapped* by adding latches between these buffers. The latches are enabled with the same clock signal that is connected to the start of the delay line, and thus can show how far the clock can propagate through the buffers within the time the latches are enabled, i.e., half a clock cycle.

When any other circuit on the same PDN becomes active, power is consumed, leading to a voltage drop that slows down the buffers of the delay line, resulting in a reduced numeric value in the TDC's output register. As this unary value can be quite large (i.e., 64 bins in our case), a priority encoder is used to reduce this data to 6 bits. Because of a symptom that higher valued bins can sometimes be faster than lower valued bins, *bubble correction* needs to be applied [59].

To save area of the sensor, usually only the last bits of the buffer chain are tapped, as the delay of the buffers do not change enough to affect the complete delay line. Thus, Figure 2.2 shows part of the delay chain to be *observable* and another part to be the *initial* delay. In FPGAs, the observable part is usually implemented using carry-chain primitives (for Xilinx: CARRY4), as they provide the finest resolution per bit. However, the initial delay is then based on elements with less area overhead, but higher delay, like multiple LUT and latch elements, typically available in any FPGA. We show an example floorplan of this sensor in Figure 5.2.

Because the real delay of the elements used for the sensor are not known at design time, it has to be ensured that the clock signal reaches the latches through the observable delay line within the respective clock cycle. In addition, it should not reach to the last latches resulting in an output value saturated at the maximum. They need to be calibrated at runtime or through adjusting and re-mapping the design. Thus, either the delay line's length has to be adjusted to the right length, or a second phase-shifted clock has to be used on the latches. In this work, we basically adjusted the initial delay, sufficient for our proof-of-concept. Please note, a more sophisticated attacker would very well be able to use a combination of phase shift and initial delay adjustment.

What is missing in [44] and [18] is an evaluation on how the length of the initial delay will impact the time quantization, i.e. how much time each individual bit in the observable delay line represents, and in effect how detailed a voltage drop will be visible. It is also not discussed how this initial delay can be found, when not using phase-shifted clocks.

The primitives used in the observable delay line have their own delay, but the fluctuations they show are those of the entire delay line (initial and observable) until the respective latch. Thus, the higher the initial delay is in relation to the observable de-

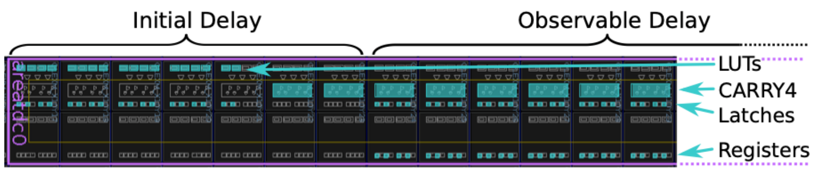


Figure 5.2.: Floorplan (rotated right) of one TDC Sensor with 18×(LUT, Latch) as part of the Initial Delay.

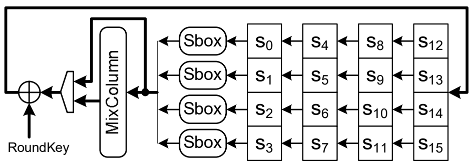


Figure 5.3.: Architecture of the underlying AES encryption core (ShiftRows and KeySchedule not shown)

lay, the more variation is zoomed into by the observable part. Thus, more fine-grained quantization levels are seen with higher initial delay, when checking the peak-to-peak variation of a given voltage fluctuation. In Table 5.1, we show the initial delay and resulting variations observed in our experiments.

5.1.2. Implementation of the PDN Trojan

To demonstrate the effectiveness of the internal sensors, we show a successful side-channel attack on an AES-128 implementation using our sensors, and compare it to an attack based on external power measurement. We first start by explaining the AES module and the target platform as well as our sensor and its properties in the following.

5.1.2.1. AES module

The AES module is a relatively small implementation with a 32-bit datapath, occupying 265 Flip-Flops and 862 LUTs. The 128-bit plaintext, after being XORed with the first roundkey, is loaded into the state registers s_i. At every cipher round, which takes five clock cycles, first ShiftRows is performed. Afterwards, as shown in Figure 5.3, at each clock cycle, four Sboxes followed by a MixColumn and AddRoundKey are performed while the state register is shifted column-wise. The four Sbox instances are shared with the (not shown) KeySchedule unit while ShiftRows is being performed. By bypassing the MixColumns during the last cipher round — in total after 50 clock cycles — the ciphertext is taken from the state register.

This AES module should generate much less voltage drop than seen in [18], since its footprint in this FPGA is only 0.3% of the total flip flops and 0.9% LUTs, versus 8% of flip flops in [18]. However, we show in the following that we can still gather sufficient information for the attack.

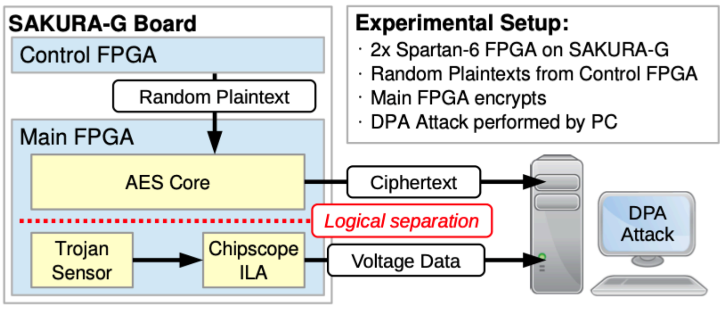


Figure 5.4.: Experimental setup showing the Sakura-G Board connected to our measurement PC, with Chipscope ILA used for data acquisition.

5.1.2.2. Target Platform and Implementation

Figure 5.4 gives an overview of our experimental setup. We ran our experiments on the widely-used side-channel evaluation platform SAKURA-G, featuring a *main* and a *control* Xilinx Spartan-6 FPGA. The main FPGA is a larger XC6SLX75 for security implementations, controlled by an auxiliary Spartan-6 XC6SLX9. As a proof of concept, we considered the aforementioned AES encryption module as the targeted cryptographic core, implemented in the main FPGA and ran at a frequency of 24 MHz. The control FPGA generates random plaintexts to be encrypted on the main FPGA. Our Trojan circuit to measure voltage sits in the main FPGA, logically disconnected from the AES module. When the AES module sends out the ciphertext, we also receive the voltage data from our sensors on the workstation, by utilizing the Xilinx Chipscope Integrated Logic Analyzer (ILA). Here, the sensor values are first stored in the internal BRAM and are then read out using the JTAG interface.

In Figure 5.5 (left), we show the entire floorplan of the experimental setup. We only place our design in the lower part of the Spartan-6. In the center region, the AES core is fixed and the sensor is placed on the left side of the AES module. The FPGA slices used for the sensor's delay line, including latches and output register, are not shared with any other logic. For all the experiments, we kept the same placed and routed partition for the AES core, in order to keep the results comparable. However, the logic required for the ILA core are automatically added each time by the synthesis tool.

5.1.2.3. Data Acquisition

We compare the efficiency of our developed sensor to a traditional measurement setup. To this end, we measured the voltage drop over a $\approx 0\,\Omega$ shunt resistor[1] in the Vdd path using a Picoscope 6403. Figure 5.6 (top) depicts the resulting trace, measured at 625 MS/s showing approximately 120 quantization levels. Note that the round structure of the underlying AES implementation can be observed through ten similar patterns, each including five smaller peaks of each individual step, respectively. A 24 MHz clock is externally given to the main FPGA which supplies both the AES module and

[1]The built-in shunt resistor of the SAKURA was shorted with a jumper.

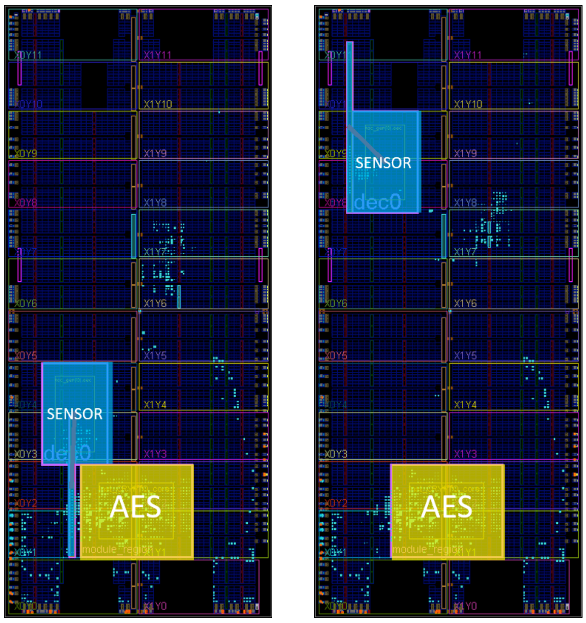


Figure 5.5.: Floorplans showing the Experimental Setup with all the relevant parts. **Left:** the internal sensor is placed close to the AES module. **Right:** the internal sensor is placed far away from the AES.

our developed sensor. Thus, in contrast to the oscilloscope that has an independent time base, the internal sensor can sample the power consumption synchronously. The side-channel information is expected to be amplitude-modulated over the clock signal, i.e., it is visible at the clock peaks. Therefore, it would be enough to sample the power consumption (only) at this exact moment when the side-channel information leakage occurs. This drastically lowers the required sample frequency for a successful attack [130]. To verify this, we conducted different experiments by supplying the sensor with different frequencies (24 MHz, 48 MHz, 72 MHz, and 96 MHz) while the AES module always runs at 24 MHz. To this end, we used a Digital Clock Manager (DCM) to generate the desired clock frequencies based on the external 24 MHz clock.

5.1.2.4. Sensor Feasibility Discussion

In our experiments we used Xilinx Chipscope to read the sensor values. Note that besides using the same clock source, there is no connection made between the AES core and the sensors, or the logic belonging to Chipscope. However, our developed sensor has the additional advantage that even if it is not synchronized with the AES clock, it catches all the variation that occurs in half of each clock cycle, since the clock traverses the delay chain during half of the clock period in which the latches are enabled (see Section 5.1.1.2).

When we use a 180° phase-shifted clock (or negative latch-enable signal) for either the latches or a complete second sensor, the average of all variation in the time between

Table 5.1.: Overview of different sensor's sampling frequency with AES module @ 24 MHz.

Sampling frequency (MHz)	96	72	48	24
No. of primitives used for initial delay	10	14	22	46
Observed peak-to-peak variation	6	6	8	15

two samples can be covered. This is an advantage over oscilloscope based samples, so even when the sensors clock domain would be separated, enough information can be inferred.

Although we use JTAG to connect to Chipscope in our experimental setup, an actual attacker would easily be able to use whatever remote connection he has, to transmit the sensor values from internal BRAM to the outside. Since no logical signaling between the attacked module and the sensor is desired, the attacker would need to adjust a mechanism to trigger the start of saving the samples e.g., into the BRAM. This can be achieved by observing the measured signal itself and trigger by detecting a large peak. Indeed, the sensor value varies only slightly, indicating that the AES is inactive (cf. Figure 5.6). The power consumption of the first round of the AES module results in a large negative peak in the sensor value, enabling a stable reference point for aligning the traces.

As described in Section 5.1.1.2, for each sensor frequency, the initial delay of the sensor has to be adjusted. This leads to different levels of quantization, and thus the observed peak-to-peak variation. This relationship is verified by our experimental data in Figure 5.6, where sensors at lower operating frequencies show higher peak-to-peak variation (cf. Table 5.1).

5.1.3. Results

In the following, we provide experimental results showing a successful attack using the traces measured by the internal sensor. We compare the results to a traditional measurement setup, i.e., measuring the power consumption externally.

As an example, we use a standard CPA attack on the AES module. Only a few bits within each byte of the internal state showed a strong leakage. Hence, we chose to predict only a single bit b to evaluate our key hypothesis k_{hyp}. Note that this was identical both for the oscilloscope as well as the internal sensor. We ran the attack on all bits of the state. The results in the following correspond to the bit position *bitpos* showing the highest correlation. We have chosen the state just before the SubBytes operation at the last round. Based on a ciphertext byte c_i, our model is

$$b = \mathsf{Sbox}^{-1}\left(k_{hyp} \oplus c_i\right) \wedge (2^{bitpos}).$$

5.1.3.1. Sensor placed close to the AES core

Figure 5.7 depicts the results using the oscilloscope as well as placing the internal sensor close to the AES core, with a gap of just 4 FPGA slices to avoid potential crosstalk. In all cases the correlation curves using 5 000 traces and the progressive curves over

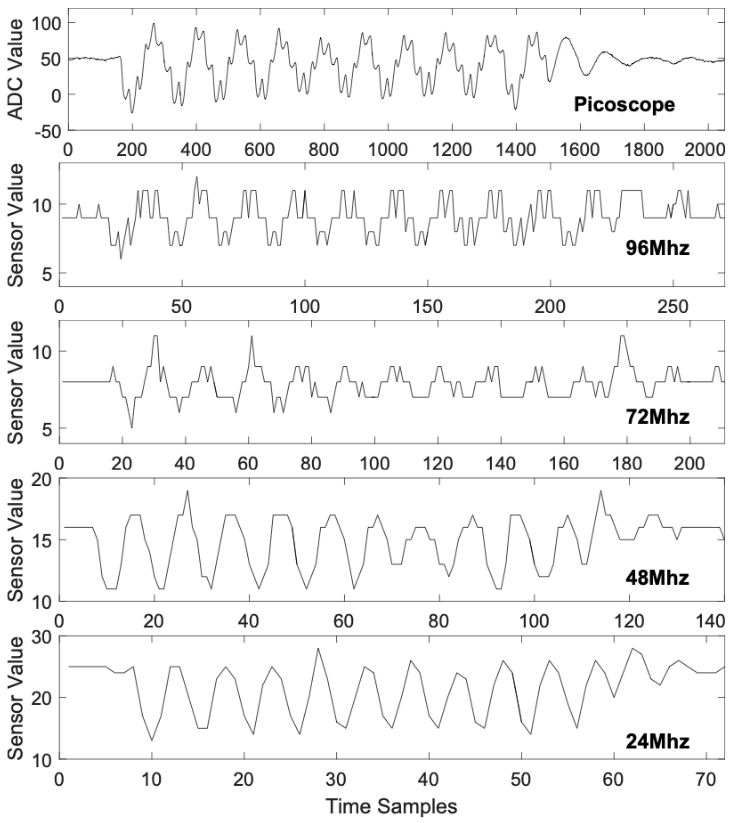


Figure 5.6.: Single traces measured using an oscilloscope (top) and using our developed sensor at different sampling frequencies (below). Time samples refer to the individual samples captured at the respective sampling rate.

the number of traces are shown. Starting with the result using the oscilloscope, we observed the maximum correlation of approximately -0.3 for the correct key hypothesis. As shown, the attacks using the internally-measured traces by the sensor are also successful. The correct key hypothesis is clearly distinguished from the others, but with a slightly lower maximum correlation of about -0.2.

Comparing the results of the sensor at different sampling frequencies, we do not observe a large deviation. This is caused by the synchronous sampling as most of the information is contained in the respective peak anyway. Finally, we can observe that the higher resolution (more quantization steps) slightly improves the maximum correlation.

5.1.3.2. Distant Sensor

We further investigated whether we still can detect any side-channel leakage in case the sensor is placed far away from the cryptographic block. We placed the sensor in

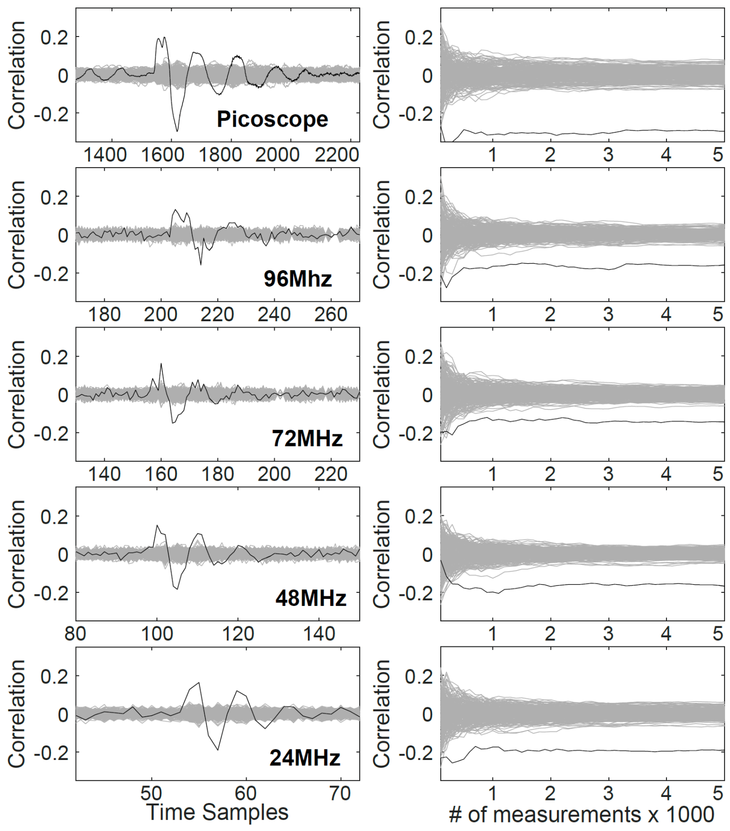


Figure 5.7.: Results using the oscilloscope (top row), using the internal sensor at different sampling frequencies (rows below), for each the correlation by means of 5 000 traces (left) and the progressive curves over the number of traces (right). The correct key hypothesis is marked in black. Time samples refer to the individual samples captured at the respective sampling rate.

the opposite region as far away as possible from the AES module. The right part of Figure 5.5 depicts the corresponding layout. We examined this situation only with 96 MHz sampling frequency, i.e., the worst case in Figure 5.7. The corresponding CPA results are depicted in Figure 5.8, indicating that the successful attack is still possible with only a slight decrease in the correlation. This highlights the high risks involved when sharing an FPGA among multiple users. Note that for a real-world design, additional logic might be placed in between the AES and the sensor, resulting in noise and an increased number of required traces for a successful attack. Anyhow, such effects are also present for an external measurement. As stated, for the presented results we made use of a SAKURA-G board optimized for SCA evaluations. However, we were able to collect similar traces on standard Artix-7 and Zynq-7000 FPGA evaluation boards as well.

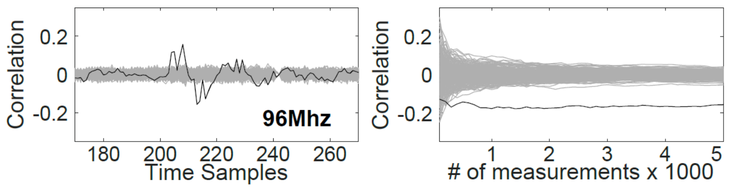


Figure 5.8.: Correlation using 5 000 traces (left) and progress of the maximum correlation over the number of traces (right) using the internal sensor at 96 MHz sampling frequency, placed far away from the AES module.

5.1.4. Extended Analysis inside a Lattice ECP5 FPGA

This is an analysis carried out for [25], in which a Lattice ECP5 FPGA was also tested, whether an on-chip power analysis attacks are feasible.

We show that Lattice FPGAs are similarly vulnerable to a CPA attack, demonstrated on a Lattice Semiconductor ECP5 FPGA on the Lattice ECP5 evaluation board (LFE5UM5G-85F-EVN) with TDC-based on-chip sensors. To implement a sensor delay line as described in Section 2.5 we use low-level ECP5 carry chain primitives (*CCU2C*), in which one FPGA slice is required for each 2-bits of the TDC sensor.

Using that sensor, we measure the on-chip fluctuations during the time a hardware AES module implemented in the FPGA is performing encryption, for 10,000 random plaintext messages. To perform the key recovery attack, the encrypted plaintexts (ciphertexts) and voltage fluctuation data from the TDC are available. We target the secret key in the last round of the AES encryption, correlating the sensor traces with a standard Hamming-Distance for each byte based power model as explained earlier in Section 2.8.

Figure 5.9 depicts the exemplary result of the successful CPA for a single byte. The correct key byte, which is marked red, shows a significantly higher correlation in the CPA results than the other 255 key candidates, shown as grey plots. The correct key byte can be identified once it is clearly distinguishable from the others. With this simple attack, approximately 50% key bytes can be recovered, with up to $100,000$ traces, recorded in less than three hours. This result proves the fundamental vulnerability to such attacks. More sophisticated analysis or more traces can recover the full key.

5.1.5. Section Conclusion

We have shown a Trojan which exploits the PDN as side channel to successfully retrieve secret keys. To this end, we have developed sensors using reconfigurable resources of FPGAs to internally capture the dynamic power consumption. We analyzed the feasibility to sense minor variations through sufficient quantization. This relies on the characteristics of the PDN of an FPGA: When the FPGA logic toggles, the supply voltage fluctuates, which is observable through the PDN. The calibrated delay sensors allow inferring the power consumption indirectly from delay changes due to such voltage fluctuations.

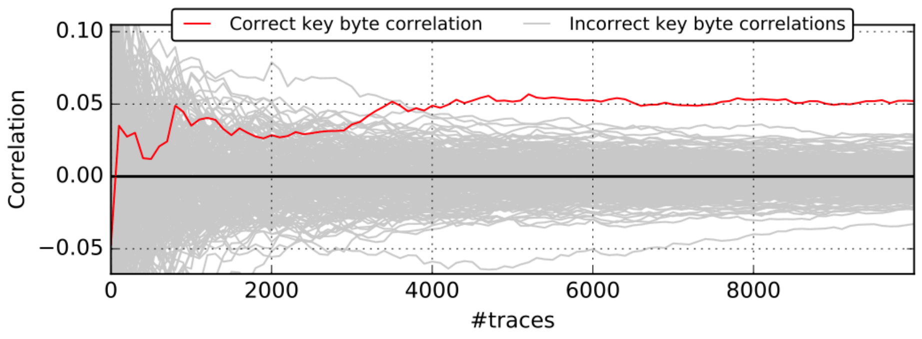


Figure 5.9.: CPA attack through on-chip sensors on the Lattice ECP5 FPGA; Correlation progress over 10000 samples for all 256 secret AES key byte candidates with the correct key byte marked red.

The Trojan can be inserted remotely without requiring physical access and with no signal connection to the attacked module. Further, it provides a very strong side channel to the entire device, even if the sensor is not placed in proximity of the attacked module. In fact, our work is a proof of concept and warns that even with 100% logical separation between the modules, the PDN carries SCA information which makes many security threats and attacks possible. This reveals a major vulnerability in emerging applications of FPGAs, such as FPGA fabric being shared between multiple users. While we have used an FPGA for our experiments, this type of attack can be transferred to other ICs and SoCs as well.

5.2. Remote inter-chip power analysis side-channel attacks at board-level

Board-level integration is a complex engineering challenge in which many components from various vendors, geographically scattered in the world, are integrated into a single PCB to form a bigger electronic system [131]. To verify the system, a fully trusted supply chain is assumed and important for system safety and security [132]. Leaving aside the difficulties of establishing such a trusted supply chain, the chips in such larger systems are based on their own software or firmware images that might be supplied independently from the chip itself. Even after manufacturing and integration, some chips are updated throughout their operational lifetime, while the system is in use. In an existing chain of trust, these risks are mitigated by cryptographically signed firmware updates [133], as they are supplied by manufacturers of smart TVs, networking equipment, routers and any sort of IoT or home appliances, just to name a few.

These chains of trust can be broken in many cases. First of all, outdated cryptography might be used for trusted firmware upgrades that is easy to break due to being legacy or is not protected against all types of attacks [10, 134]. In other situations, the manufacturer secret keys can be extracted or get leaked, or certain devices in the system might not support the mechanisms required for trusted firmware updates at all [135].

Full system integrity is also hard to guarantee if software or firmware from a 3rd party is run on any chip in the system. In these situations, malicious applications can be introduced accidentally, for instance by executing any content from the Internet, which might just be javascript on a website [96]. In those cases, it is of high importance to provide proper isolation of individual system components, typically handled at the logical level (cf. javascript sandboxes). For instance, in the recent Meltdown and Spectre [11, 12] attacks it has been shown that such isolations can be broken, and a user can escalate their privileges and gain superuser access.

Particularly FPGAs are increasingly used as accelerators in many systems, ranging from Cloud-computing appliances [2–4] to getting integrated in complex SoC devices. Very small FPGAs are often inserted as *glue-logic* as a translation layer between other existing devices. What all FPGAs in these use cases have in common is that they are part of a bigger system, probably sharing the power supply with other components, i.e. even as a PCIe device.

As the previous section and [105] have shown, a chip containing FPGA fabric can be used to implement sensors that are sufficient for *remote* power analysis side-channel attacks within the FPGA. This threat from power analysis attacks was previously assumed to require an attacker with physical access.

In this section, we escalate the risk of remote power analysis attacks from the chip to the board-level, affecting much more potential components of an entire system. We show that even through multiple levels of a power distribution network, in which capacitive, inductive, and resistive effects persist, sufficient side-channel information can be extracted to attack a cryptographic module in another chip placed on the same board.

Contributions: Our main contribution is a first proof of a board-level SCA attack from one chip to another, based on software-reconfigurable firmware. In short, the contributions of this work can be summarized as follows:

- Our results prove that board-level power analysis attacks are possible through firmware, and also highlight the threat of a malicious chip introduced in the supply chain.
- Depending on the system configuration, the attack can be introduced remotely and in software,
- We provide two case studies on an inter-chip attack on AES and RSA, proving the high risk of this threat.
- If local access to a system is given only for a short time, the attack can infect a system in a covert way, because no external or dedicated measurement equipment is required.

Outline: The rest of this section is organized as follows: Section 5.2.1 elaborates our adversary model in more detail and explains some background knowledge on board-level supplies and power-analysis side channels. In Section 5.2.2, our experimental setup will be explained, followed by our results given in Section 5.2.3. Finally, we conclude on these results in Section 5.2.4.

5.2.1. Adversary Model and Threat Analysis

Secure system design typically involves a trusted computing architecture in which adversaries are allowed to manipulate any chip outside of a Trusted Computing Base (TCB) in order to attack it by any means possible [136]. The TCB is achieved through *isolation* and *attestation*. Isolation is implemented by access control to and from secure enclaves, and attestation proves the integrity of system components within the enclaved TCB, for example through protecting firmware images against unauthorized modification. Because most trusted computing models do not consider the electrical level, a famous architecture for SoCs, ARM TrustZone, was shown vulnerable to fault attacks through power management [137].

While previous chapters have purely addressed on-chip attacks, here we assume related threats, elevated to the board-integration level. In Figure 5.10 we show an adversary scenario on a PCB with multiple chips, supplied by the same power supply. Two of them (Chip A and B) are within a secure enclaved TCB. Chips outside of the TCB can be accessed locally or remotely by a 3rd party entity, which can be an attacker in Chip C. The goal of the adversary is to gather secret information from a victim component within the secure enclave, for instance a cryptographic accelerator in Chip A.

There are two possible cases for the adversarial access to Chip C outside of the TCB:

1. Chip C is provided by an attacker which can access the supply chain to introduce a malicious chip into the system.

2. Chip C is a benign chip by design but can run different software or firmware (for instance, a cloud accelerator). The adversary is restricted to reprogram the single device *Chip C*, which is logically isolated from the enclave, but shares the same power supply with the victim in the enclave.

By means of measurements on the power supply, Chip C can thus attack Chip A in the enclave, even when a TCB was logically established. After the attack, the adversary

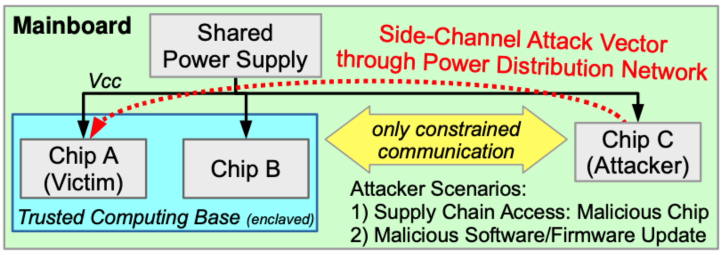

Mainboard
Shared Power Supply
Side-Channel Attack Vector through Power Distribution Network
Vcc
Chip A (Victim)
Chip B
Trusted Computing Base (enclaved)
only constrained communication
Chip C (Attacker)
Attacker Scenarios:
1) Supply Chain Access: Malicious Chip
2) Malicious Software/Firmware Update

Figure 5.10.: Scenario of a shared power supply leading to a risk of side-channel attacks. In this example Chip C tries to deduce information from the power supply on Chip A.

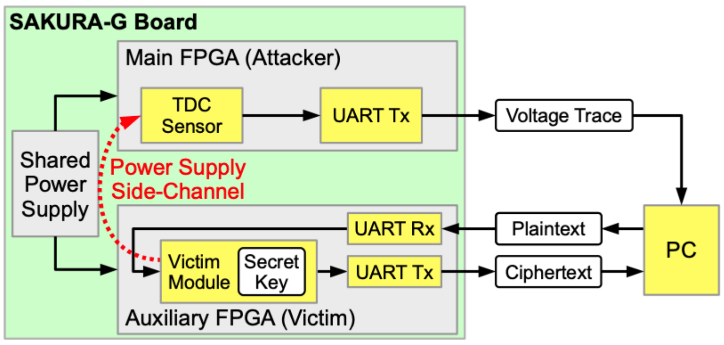


Figure 5.11.: Experimental setup showing the SAKURA-G Board connected to our measurement PC.

with access to Chip C can use any type of communication channel to transmit the side-channel information remotely and analyze it, to extract secret keys from the victim system. If no proper communication channel exists, covert channels can be used to transmit this information [126, 138, 139].

5.2.2. Experimental Setup

In this section we explain our experimental setup in general and then go on to briefly explain the victim and attacker designs. Section 5.1 has shown that SCA attacks are possible within a chip. Beyond that, we prove here that such attacks can be performed inter-chip on the board-level. To this end, we use the well-known SAKURA-G board [140] that was designed for external side-channel analysis research, e.g., by offering measurement points for traditional oscilloscopes.

The main reason for using the SAKURA-G in our experiments is that it contains two FPGAs on a single board, where both can be freely programmed. It contains two Spartan-6 FPGAs: a small *auxiliary* FPGA (XC6SLX9), and a larger *main* FPGA (XC6SLX75). We ran different configurations for the capacitors between VDD and GND, i.e., the unaltered default configuration of the SAKURA-G and with all small-value capacitors close to the FPGAs removed. In any case, we inserted a small bridge so that the core voltage of the main FPGA is provided by the same power supply as for the auxiliary FPGA. This modification does resemble more typical industrial boards, in which power supplies of the same voltage level are shared for efficiency reasons.

Figure 5.11 gives an overview of this experimental setup. The auxiliary FPGA receives plaintexts from the PC and encrypts them with an internal secret key, sending the ciphertext back to the PC. The main FPGA uses a sensor and transfers the sensor data to the PC. For the ease of experimentation, the main FPGA also receives a trigger signal whenever the auxiliary FPGA starts an encryption. In a real-world scenario, an encryption might be triggered externally (by the PC) or the traces can be realigned using existing work. On the PC, we then launch simple and differential power analysis attacks on the recorded sensor data. Fig. 5.12 shows the two FPGAs on the SAKURA-G Board and their respective roles in our setup.

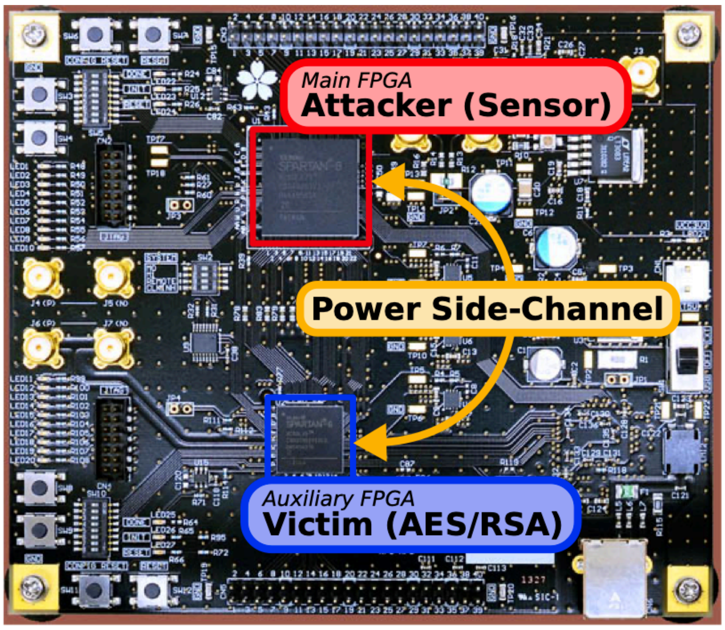


Figure 5.12.: Setup showing the configuration of the two FPGAs on the SAKURA-G board. The sensor to attack is in the main FPGA, while the cryptographic module (AES or RSA) runs on the auxiliary FPGA.

Our victim designs are implemented on the smaller auxiliary FPGA. This FPGA is still large enough to fit an AES module and a small RSA implementation, both described in the following.

5.2.2.1. AES victim module

The AES module we use here is a simple implementation that is not side-channel protected and follows the same design principle that was explained in Section 5.1, to be comparable. On systems that are just remotely-accessed, considering such an unprotected module is a valid assumption, since the threat of power analysis side-channel attacks is not considered in remote adversary models. We use a 128-bit AES implementation that is based on a 32-bit datapath. The 128-bit plaintext is XORed with the first roundkey and then loaded in the state register s_i. Each cipher round requires 5 cycles in this implementation. In each subsequent cipher round, the respective operations for AES, Byte Substitution in the Sbox, ShiftRows (not shown since it is only re-wiring), MixColumn and AddRoundKey are performed. In total the encryption takes 50 clock cycles and the resulting ciphertext can be acquired from the state registers. In the used auxiliary FPGA (Spartan-6 XC6SLX9), the resource utilization is minimal, but percentage-wise higher compared to the larger FPGA used in Section 5.1. Here ,the AES module is also running at 24 MHz.

Algorithm 1 Right-to-left binary exponentiation (cf. 14.76 of [141])

```
Input: Message x, Exponent e, Modulus N
Output: $x^e \mod n$
 1: $A \leftarrow 1, S \leftarrow x$
 2: while $e \neq 0$ do
 3:   if $e$ is odd then
 4:     $A \leftarrow A \cdot S \mod N$
 5:   else
 6:     $A \leftarrow A \cdot 1 \mod N$
 7:   end if
 8:   $e \leftarrow \lfloor e/2 \rfloor$
 9:   $S \leftarrow S \cdot S \mod N$
10: end while
11: Return($A$)
```

5.2.2.2. RSA victim module

We implemented RSA using a straightforward right-to-left binary exponentiation [141], i.e., following the same principles as the one used in [105] for the sake of comparability. The pseudo-code is given in Alg. 1. In each iteration step of the exponentiation, a squaring is executed. If the current (secret) bit of the exponent is set, the squared term of the previous step is multiplied to the register storing the result.

Like [105], if a specific step of the algorithm does not require a multiplication, one of the inputs is set to 1, i.e., calculating the identity function. Thus, in order to retrieve the secret exponent, the adversary tries to identify whether or not an actual multiplication took place for each of the steps in the binary exponentiation.

Both squaring and multiplication are implemented as separate modules using dedicated multiplication cores with integrated modular reduction. The multipliers itself operate on the shift-and-add principle: In each clock cycle, one operand is multiplied by two (left shift) and reduced. If the current bit of the other operand is set, the shifted term gets added to the result register.

The limited resources of our auxiliary FPGA only sufficed for a rather small key-size of 224-bit, running at 24 MHz. However, we stress that RSA implementations with a larger key size are usually much easier to attack using Simple Power Analysis (SPA) than smaller ones. This stems from the fact that because of the larger operand sizes, the multiplication cores require more cycles while consuming more power as well.

For our proof-of-concept implementation, we have 224 steps of the binary exponentiation, each requiring 224 clock cycles for the squaring and the multiplication running in parallel. Thus, for each step we have 224 clock cycles for which we have to decide whether a multiplication is taking place or not. Considering for example an RSA with a 4096-bit key size, the computation time is increased to 4096 cycles likewise.

5.2.2.3. Sensor Attacker Module

We use the same TDC sensor from Section 5.1 to measure a fraction of the power supply noise of the system, but here only look into measuring inter-chip attacks. We also made sure to run our sensor in the same frequencies as in the previous work of 24, 48, 72, and 96 MHz. As we also use the same type of evaluation board and FPGAs, this allows for comparable results. Furthermore, these frequencies are easily created based on the board-level 48 MHz crystal oscillator.

We transmit our data through UART in our experimental setup. In a real design other communication channels might be required. Thus, part of the key recovery algorithm itself could even be performed on the FPGA, to transmit the data in a more compressed format later on. If the compromised chip has no access to communication devices, it might need to send information through one of the numerous possible covert channels that previous works have explored [126, 138, 139].

5.2.3. Results

In the following, we present experimental results attacking AES using CPA and RSA using Simple Power Analysis (SPA). For both cases, the respective cipher was running on the auxiliary FPGA while the TDC sensor captured the inter-chip voltage drop through its supply pin on the main FPGA.

5.2.3.1. Attack on AES

As discussed in Section 5.1, the TDC sensors incorporate an inverse relation between the sampling frequency and the amount of variation that is captured. Hence, we chose to run the attack for different sampling frequencies as well. Figure 5.13 depicts the exemplary traces, for each sampling frequency averaged over 1000 measurements. The AES encryption is taking place approximately between 3.6 and 5.7 µs.

Similar to Section 5.1, we run a textbook CPA attack, using the ciphertexts to predict the state before the last Sbox operation based on key hypothesis. Figure 5.14 depicts the corresponding results for the attacks: For each sensor frequency, the correlation after 500 000 traces is plotted on the left and the progress of the maximum of the correlation on the right. The curve belonging to the correct key candidate is marked in black. Indeed, the attack succeeds for all tested sampling frequencies, yet at a large deviation relation to the amount of required traces. For the 96 MHz sensor, the correct key candidate is starting to stand out after approximately 200 000 processed traces. Considering the 24 MHz sensor instead, the correct key candidate is visible immediately after around 20 000 traces.

In Section 5.1 the sensor and target were implemented within the same FPGA If we compare the results from here with those, we require more traces by a factor of 40, considering the respective best attacks. It should be noted that none of the smallest-value capacitors between VDD and GND were put in place when measuring the inter-chip leakage. Only distant large-value capacitors were left in the VDD path as such do not affect small variations anyway. Anyhow, we ran additional experiments with the default configuration of the capacitors on the SAKURA-G, i.e., all small capacitors

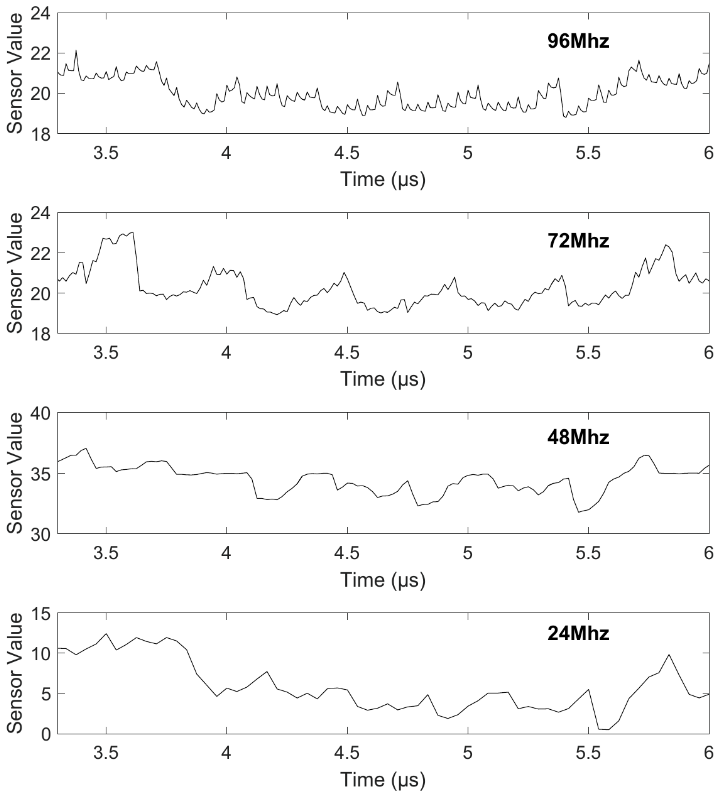


Figure 5.13.: Averaged traces measured during AES using the TDC sensors at different sampling frequencies.

close to the FPGA chip were placed. A direct comparison at the sampling frequency of 24 MHz is depicted in Figure 5.15. As expected, such additional capacitance acts as a low pass filter that can be compensated by increasing the number of captured traces. Indeed, when sampling at 24 MHz the correct key candidate started to stand out after approximately 2.5 million traces using the default configuration but powering through the same power supply. Note that 2.5 million traces still only correspond to around 38 Mbyte of encrypted data when using AES-128.

5.2.3.2. Attack on RSA

Based on the results for AES, we chose to measure the RSA core using a sampling rate of 24 MHz. As before, both FPGAs share a power supply and all capacitors are in place. The RSA is running at 24 MHz. Thus, we require at least 50 176 cycles to capture the whole binary exponentiation (224 clock cycles for each of the 224 steps). Figure 5.16 depicts the raw trace with an already visible variation over a time span of approximately 2100 µs.

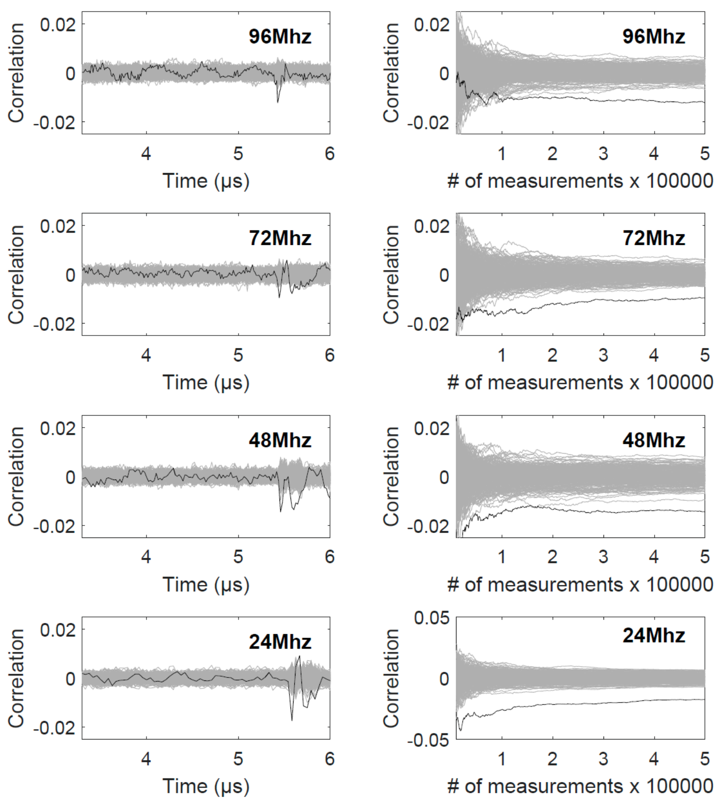


Figure 5.14.: CPA attack on AES: Results to estimate the sensor quality at different sampling rates, with a board when all relevant capacitors are removed. Each row shows the correlation using 500 000 traces (left) and the progressive curves over the number of traces (right). The correct key hypothesis is marked in black. Time samples refer to the individual samples captured at the respective sampling rate.

We recall that the adversary's goal is to recover the secret exponent by identifying whether the multiplication took place or not. Every time a multiplication is performed (in parallel to a squaring), the circuit consumes more power. Figure 5.17 depicts a detailed view after applying a low pass filter with a cut-off frequency of 900 kHz. Instead of simply capturing the increased power consumption during the multiplication, we can observe that the TDC sensor receives a differential signal of the encryption. Thus, we have to consider three different cases how the conditional multiplication in the binary exponentiation will affect the TDC sensor:

- If the multiplication module is switched from the off-state (factor of 1 applied to an input) to the enabled-state, the voltage will briefly drop until the power supply is compensated for the increased current. The voltage drop will slow down the signal in the TDC sensor, leading to a negative peak in the trace. An arrow pointing downwards indicates this case is Figure 5.17.

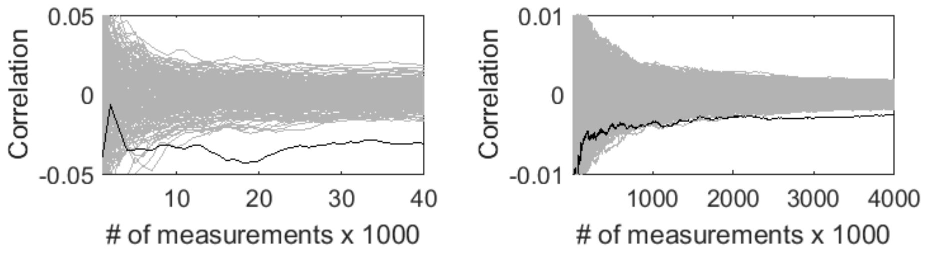


Figure 5.15.: CPA attack on AES: Progressive curves over the number of traces with a board when all relevant capacitors are removed (left) and with the default capacitor configuration (right), both sampled at 24 MHz. The correct key hypothesis is marked in black. Time samples refer to the individual samples captured at the respective sampling rate.

- In case the multiplication is switched from the on-state to the off-state, i.e., the FPGA suddenly consuming less power, the voltage overshoots briefly until compensated. This leads to a positive peak in the trace due to the accelerated sensor. This case is marked using an arrow pointing upwards. Also note the very large positive peak at the end of the exponentiation in Figure 5.16, indicating that both the multiplier and the squaring module got deactivated.
- If the state of the multiplication does not change, i.e., either staying enabled or staying off, the power consumption remains identical. Thus, the voltage level is constant, causing a steady sensor value. This is indicated by a dash.

These three cases are marked in the magnified view of Figure 5.17. Indeed, the secret exponent can be read out easily even though the RSA and the TDC sensor are implemented on separate FPGAs sharing the same source of supply voltage.

5.2.3.3. Discussion

Our results prove that board-level power analysis side-channel attack threats exist, even in the presence of decoupling capacitors. Of course, same or even better results can be achieved when adding a dedicated ADC directly to the power rails, but obviously not without raising questions of its use. Instead, a seemingly disconnected FPGA would not raise any alarm even in a fully trusted supply chain. The malicious behavior can be enabled later on by a firmware to measure the supply voltage with the sensors.

Note that this threat is not limited to FPGAs in a board-level integrated system when full supply chain trust is not ensured. Instead, such a threat exists for any untrusted chip on the board. For example, an attacker could use an undocumented internal ADC connected to the shared power supply as a power analysis attack vector. The same is true for any other chip on the board that can be used as a measurement device through maliciously altered firmware. An increasing number of sensors are integrated into all kinds of chips for increased reliability and monitoring purposes, even for voltage fluctuations [56, 142], elevating the risk of maliciously measuring them for power analysis attacks. In a system where only remote access is considered to be

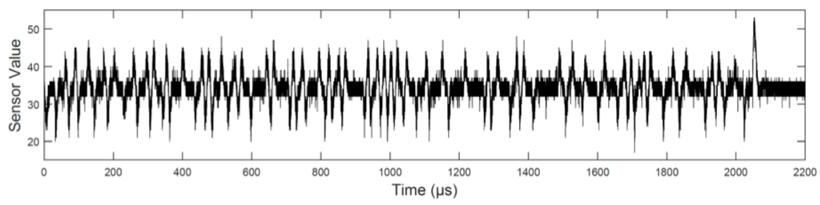


Figure 5.16.: Binary exponentiation for RSA captured with the TDC sensor on separate FPGAs, sampled at 24 MHz (raw trace).

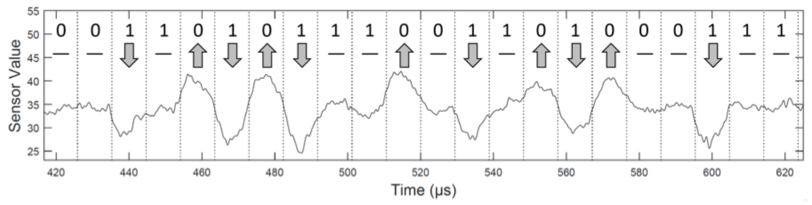


Figure 5.17.: Detail of the binary exponentiation captured with the TDC sensor after applying a 900 kHz low-pass filter. Dotted lines mark the time-span of an individual step in the binary exponentiation. Arrows indicated whether the state of the multiplication module changed (on to off: arrow upwards, off to on: arrow downwards, and no change: dash). The bits above are a part of the (correctly) recovered secret exponent according to this classification.

an attack vector, integrated cryptographic accelerators are often not protected against power analysis side-channels (i.e., only timing side-channels are avoided). Such systems could thus be attacked remotely if proper electrical isolation on the board integration level does not exist.

5.2.4. Section Conclusion

In this section we showed the feasibility of launching remote power analysis attacks on a full board, in which multiple trusted and untrusted chips share the same power supply. Based on an exemplary FPGA, we show the high risk involved in integrating chips on a board whose firmware or configuration is unknown at the time of board-integration. We integrate sensors that are known to measure the voltage fluctuations inside FPGA fabric and prove that they can also capture a fraction of the external inter-chip voltage drop. Using this mechanism, *any* chip next to the FPGA can be analyzed by power side-channel analysis attacks. We prove that this mechanism can lead to a successful key-recovery attack on an AES and an RSA core in a chip neighboring to the FPGA without any logical connections.

We should emphasize that the attack presented here is not limited to FPGAs, which only serve as a case study. Other untrusted chips, through untrusted manufacturing or untrusted firmware updates in the field, might also be able to measure such voltage fluctuations from a neighboring chip, e.g. using voltage monitors that are integrated for reliability reasons, but are re-programmable to serve other purposes. Thus, these results give a warning sign to all board-integrators that proper isolation is also required on the electrical level, as even a remote attacker might get hold of sufficient sensors for inter-chip board-level power analysis attacks.

5.3. Leaky Noise: New Side-Channel Attack Vectors in Mixed-Signal IoT Devices

Traditional applications of microcontrollers and SoCs are embedded systems with dedicated and limited interfacing capabilities, which typically have to fulfill the requirements of low-cost and energy efficiency. As these devices are recently used in Internet of Things (IoT) applications, their security against remote attacks becomes a major concern [143, 144], as several breaches have already been reported [10, 145]. Unlike security for highly dependable server systems, the energy efficiency imposes lightweight cryptography [146]. However, there are potentially new security threats associated with the underlying technology, which is not widely researched yet [147].

One of these technology-dependent threats is due to their *Mixed-Signal*-integration of analog and digital logic on the same SoC. Inside the SoC, cross-coupling or voltage fluctuations caused by the digital part can bias the analog circuit [38]. This bias has been shown to even affect radio transceivers, such that electromagnetic (EM) side-channel attacks [148] can be performed in proximity of up to 10 meters [147]. One of the most widespread analog circuits that is integrated in SoCs is an ADC, essential in many complex modules and applications such as temperature sensors, wireless transceivers or multimedia audio applications, just to name a few. Even more, multiple of such systems are often connected into complete sensor networks made out of *smart sensors* [149], and in so-called Edge-IoT devices [150]. This is evident by the support in Amazon [151] and Microsoft [152] IoT cloud applications for many SoC and microcontroller platforms.

In such mixed-signal SoCs, the analog and digital subsystems typically have a common power supply, and are spatially close on the same die or package, leading to either crosstalk or voltage fluctuations from the digital logic propagating into ADC measurement results, known as one of the challenges in mixed-signal design [37].

For FPGAs, it has already been shown in Section 5.1 and in [105, 153] that it is possible to mount power analysis attacks just through software configuration. Those attacks can also be extended to other chips on the same PCB through a shared power supply, as shown in Section 5.2. For small wireless systems it has been shown that electromagnetic side-channel leakage can be observed in close proximity of a few meters to the device, without having to probe it directly [147]. Thus, there is increasing evidence that power analysis attacks, originally considered a local issue, can also be used in remote exploits.

In this section, we present another type of power analysis side channel that can be exploited through software, potentially remotely. We show that ADC noise, which is usually characterized using statistical methods [41–43], is not just statistical noise, but is correlated to the activity in the digital subsystem. To assess the capability of this side-channel, we perform leakage assessment [72, 73] on multiple platforms. Afterwards we show a successful key recovery attack on the AES. In summary, this section contains the following contributions:

- First assessment of ADC noise as a software-only power analysis side-channel, which could be used remotely.
- Elaborate leakage assessment of this side-channel on a range of systems under different conditions, evaluated on a real-world cryptographic library.
- Successful ciphertext-based key recovery on AES using ADC noise.

In the remaining section, we first explain preliminaries in Section 4.2.1 regarding our adversarial model and the essential background information, including related work. We then explain our experimental setup in Section 5.3.2. Our results are presented in Section 5.3.3, and discussed in Section 5.3.4. Finally, the section is concluded in Section 5.3.5.

5.3.1. Adversarial Model

The basic principle of our adversarial model has two variants, which are summarized in Figure 5.18. We consider a software-based model, in which an attacker has full or partial access to ADC data, and a victim which is running cryptographic code. We assume the ADC is read during the time the cryptographic code is executed, and the adversary has access to that data, either directly through another task in the system, or indirectly through a webserver hosting sensor data. Typically an ADC can be read simultaneously by using a second core, Direct Memory Access (DMA), or interrupt-driven operation, available in most microcontroller architectures.

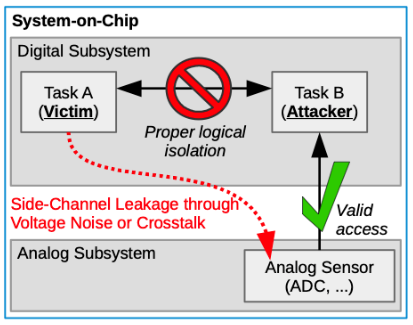
This adversarial model applies to a broad range of applications and devices. For instance, IoT applications based on smaller microcontrollers or SoCs are within that model, available from a large range of manufacturers. The *Amazon IoT FreeRTOS* project [151] has direct support for microcontroller boards from various manufacturers. Microsoft Azure also supports various systems, where many smaller mixed-signal microcontroller-based systems are included [152].

5.3.2. Experimental Setup

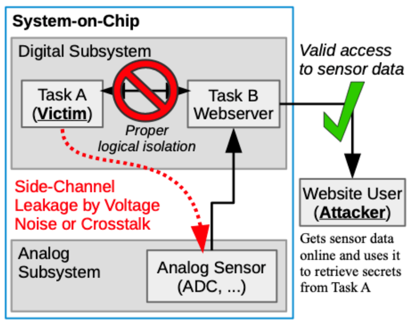
We show the overview of our common experimental setup in Figure 5.19. The basic setup consists of a set of software components and different microcontroller boards that all have basic hardware features like ADC or UART modules. We use the software stacks provided by the board vendors themselves, which is very similar across the boards. The stacks are all based on the common security library *mbedTLS* [154] and the real-time operating system *FreeRTOS* [155]. These serve merely as a readily available proof of concept platform. We do not exploit any particular vulnerabilities or side effects from this stack. In the following subsections, we explain more details of this experimental setup.

5.3.2.1. Hardware Platforms

Three different platforms are evaluated in our experiments, with the same basic experimental setup as in Figure 5.19. All of the used platforms are evaluation boards



(a) Variant of the adversarial model in which a malicious task (Task B) could gain knowledge of secret information processed in the victim (Task A), circumventing any access restrictions.



(b) Variant of the adversarial model where side-channel leakage is embedded in sensor data that leaves the system. An external attacker can then use the sensor data to retrieve secrets from Task A.

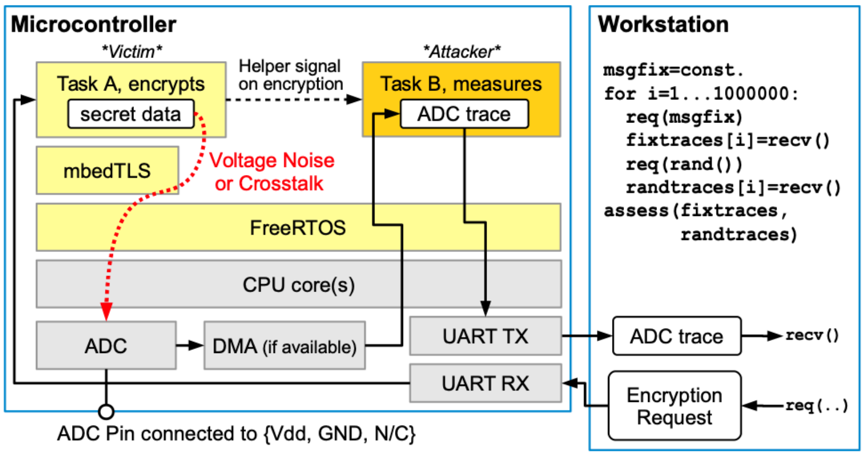
Figure 5.18.: Basic principle of the two variants of our adversarial model considered in this section. In both cases an ADC in the Analog Subsystem is biased from Task A in the digital subsystem. This bias can contain secret information that Task A processes.

for 32-bit microcontroller systems that can be used in IoT applications. These are the ESP32-devkitC, STM32L475 IoT Node, and two copies of the STM32F407VG Discovery, which were bought apart from each other. These two boards were checked in order to see how sample variation affects the results.

In the two STM32 microcontrollers, a Memory Protection Unit (MPU) is integrated to prevent operating system tasks from reading memory outside their allowed range. We did not use that unit, but it shows that these systems actually support a certain level of isolation, which could potentially be broken through the ADC noise side-channel.

All platforms run in an operating frequency range of 80-168 MHz, and respective ADC sampling frequencies were chosen, such that a whole trace of one cryptographic operation can be saved in internal SRAM memory. The ADCs of these platforms all support a 12-bit operation mode, which we selected when not noted otherwise.

The power supply on all the microcontroller boards uses the 5 V USB power as input, which we supplied from a standard PC USB output. All boards use a voltage converter to produce a 3.3 V voltage for the V_{dd} of the respective controller. In the STM32F407VG Discovery and STM32L475 IoT Node platform, the manufacturer added a compensation network of capacitors and inductors through which the 3.3 V is connected to the ADC reference pin. We did not do any modifications on any of the boards, and thus also kept this compensation network. In the ESP32-devkitC platform, only an internal ADC reference exists. The ADC of this platform can also be internally connected to Vdd, which we used throughout the results in this section for 'Vdd', instead of an external connection. The ESP32-devkitC contains three CPU cores, with two Xtensa 32-bit CPUs and a ultra-low-power (ULP) core that can run independently and also collect ADC samples. The STM32F407VG Discovery and STM32L475 IoT Node are single core platforms with Cortex-M4 CPUs, such that DMA is required to



```
msgfix=const.
for i=1...1000000:
  req(msgfix)
  fixtraces[i]=recv()
  req(rand())
  randtraces[i]=recv()
assess(fixtraces,
       randtraces)
```

Figure 5.19.: Overview of our common experimental setup, shared among the used platforms.

sample the ADC in parallel to a running CPU. This information is listed together with details on sampling in Section 5.3.2.3, Table 5.3.

5.3.2.2. Software Environment

On the software side, we use two operating system tasks in the system. One task encrypts plaintext messages received through UART from an attached w3orkstation, while another has access to ADC data. The detailed operation is explained in the following Section 5.3.2.3.

In all of the platforms, we used the development environment of the vendor and the respectively provided library versions, of which we provide an overview in Table 5.2. For the ESP32 platform, the Espressif IoT Development Framework (ESP-IDF) was used, which integrates the listed mbedTLS and FreeRTOS versions. A project is based on a simple Makefile that includes an ESP-IDF makefile. The compiler and assembler versions come from the official setup guide, and we used the default 'Release' compiler optimization for size (-Os). The STM32CubeMX software used for the ST Micorelectronics platforms is a code generator, generating Makefile-based projects, also integrating mbedTLS and FreeRTOS. The official ARM GCC compiler was taken through a package provided for the Eclipse Development Environment and we also used optimization for size (-Os). Only for a single experiment on CPA, -O0 was used, which is specifically mentioned in the results later.

Please note that STM ships an old mbedTLS with their recent development framework. For our evaluation, however, this is not relevant, since we verified that the used functions for our experiments `mbedtls_internal_aes_encrypt` and `mbedtls_mpi_exp_mod` did not contain any significant changes that would defeat the comparability of our results.

We evaluate AES-128 for single encryptions of 128-bit plaintext messages, and respective sliding window modular exponentiation with 512-bit exponentiations. For AES we use basic single encryptions with `mbedtls_internal_aes_encrypt` to prove

Table 5.2.: Used vendor toolchain versions and respective library and compiler versions

Platform	Framework	mbedTLS	FreeRTOS	Compiler(s)
Espressif ESP32-devkitC	ESP-IDF 3.1[1]	2.12.0	8.2.0 Xtensa Port[2]	xtensa gcc 5.2.0[3] esp32ulp 2.28.51[4]
ST Microelectronics STM32F407VG Discovery	STM32CubeMX[5] 4.26.1, 5.0.1	2.6.1	9.0.0	arm gcc 7.3.1[6]
ST Microelectronics STM32L475 IoT Node	STM32CubeMX[5] 4.26.1	2.6.1[7]	9.0.0	arm gcc 7.3.1[5]

[1] Espressif IoT Development Framework `https://github.com/espressif/esp-idf/`

[2] Espressif explains the Xtensa Port in `https://docs.espressif.com/projects/esp-idf/en/v3.1/api-reference/system/freertos_additions.html`, which mainly adds multicore support

[3] crosstool-ng-1.22.0-80-g6c4433a-5.2.0 as linked in `https://docs.espressif.com/projects/esp-idf/en/v3.1/get-started/linux-setup.html`

[4] v2.28.51-esp32ulp-20180809, as linked in `https://docs.espressif.com/projects/esp-idf/en/v3.1/api-guides/ulp.html`

[5] STM32CubeMX Eclipse plug in `https://www.st.com/en/development-tools/stsw-stm32095.html`, 4.26.1 was used for leakage assessment, 5.0.1 was used for the CPA attack in Section 5.3.3.5.

[6] GNU MCU Eclipse, based on arm-none-eabi-gcc 7.3.1-1.1-20180724-0637 from `https://gnu-mcu-eclipse.github.io/blog/2018/07/24/arm-none-eabi-gcc-v7-3-1-1-1-released/`

[7] For this platform, none was provided in CubeMX, but the version from STM32F407VG worked directly

principal information leakage. Please note that advanced operating modes of AES, like counter-mode, are on principle also vulnerable to power analysis [156], but require more effort. A knowledgeable attacker could deploy such attacks. For modular exponentiation, we use `mbedtls_mpi_exp_mod`, which is also used in the RSA implementation of mbedTLS, but does not prove its overall vulnerability. Please note that we only use mbedTLS such that we have cryptographic relevant code for the leakage assessment, but not to show any new attack on this specific library.

5.3.2.3. Sampling of Data, Transmission and Synchronization

As seen in Figure 5.19 in our experimental setup, a workstation PC sends encryption requests to the microcontroller system using a simple UART interface, in which one task performs encryptions, while another task has access to the ADC. To be able to record side-channel leakage from the digital to the analog subsystem, the ADC must be sampled during the time of the cryptographic operations of Task A. Task B should not acquire ADC samples while it runs time-multiplexed to Task A, since the ADC would not receive side-channel leakage. Thus, Task B needs to either run in parallel to Task A on a separate CPU, or use hardware accelerated sampling, i.e. through DMA. A separate DMA module is available in many microcontrollers or SoC systems to improve the performance of various tasks. Other controllers offer programmable state machines or specific low-power cores that can control the ADC and other peripherals in parallel to normal task execution on the main CPU(s).

In Figure 5.20 we show a more detailed description how the two tasks are executing in parallel in our experiments. More detailed example code can be found in Section B.3. At the beginning, Task A receives a message to be encrypted through UART and notifies a sleeping/waiting Task B using a FreeRTOS notification (as the helper signal). Task B starts collecting ADC data in parallel to Task A, either in dual-core operation or using DMA operation of the ADC. Task A then encrypts the message using a previously stored key, while the ADC is collecting data. After the encryption, it waits for a notification. Upon finishing the fixed number of ADC samples, Task B sends them to the workstation, and notifies Task A so everything can start afresh for another message. Due to differences in the systems, we use DMA transfer in the two STM32 systems, and dual-core operation in the ESP32. However, after we had performed all experiments, we found the ESP32 also has a sampling mode that does not require the second core, by using its i2s-module.

There is still a fundamental difference between using the ADC with DMA, versus using software on a CPU to acquire individual ADC samples in a loop. This difference is visualized in Figure 5.21. Running the ADC in a continuous mode with DMA at lower speeds will basically use more time in the ADC conversion itself, but will not reduce the total time range in which the measurement is influenced by noise. In comparison to that, in single-conversion software-based sampling, a certain time between the ADC samples will be used to run software to store data and prepare the next ADC acquisition. If we want to change the sampling rate, we will need to add delay in software. Any analog noise in that time will not affect the ADC result, and thus some side-channel leakage might not be captured in the acquired ADC data. In our experiments we also introduced delays in CPU-based sampling, because of internal memory size limitations. Since sampling with the CPU is usually slower, too, the sampling rate can also be negatively impacted by CPU-based sampling.

Usually a sampling frequency above the CPU or circuit frequency is recommended for power analysis attacks, but is not necessarily required for successful attacks [157]. For the platforms we use, an additional limitation is the amount of available internal memory to store all samples. For RSA, we typically had to sample rather slow to not fill the memory (2000-4000 samples per encryption). For AES we had to sample almost

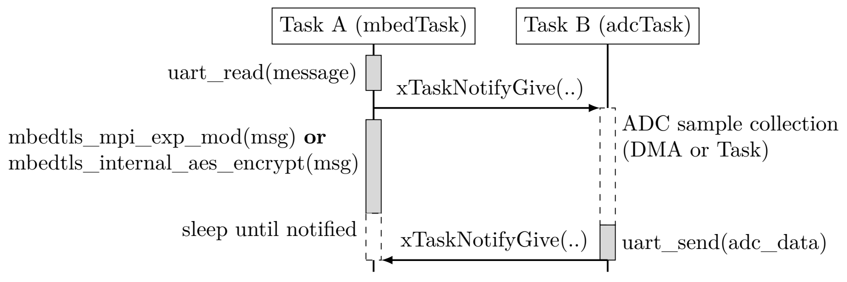


Figure 5.20.: Description of one loop iteration of the two FreeRTOS tasks.

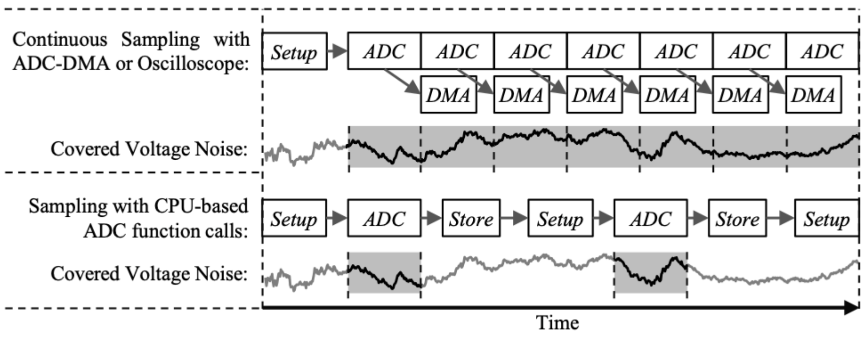

Continuous Sampling with ADC-DMA or Oscilloscope:
Setup
ADC
DMA
Covered Voltage Noise:
Sampling with CPU-based ADC function calls:
Setup
ADC
Store
Covered Voltage Noise:
Time

Figure 5.21.: Principle how different ADC sampling styles will cover less or more parts of the voltage noise affecting the ADC result. DMA needs to be used for continuous sampling, while software-based sampling (in the multi-core scenario) will always introduce some gaps.

Table 5.3.: Overview of Leakage Assessment Experiments, repeated for ADC Pin = {Vdd, GND, N/C}

Platform and Experiments	Sampling Style	ADC Ref. Filter	Algorithm	Samplerate / #Samples
ESP32-devkitC @80MHz	ULP-CPU	No	AES-128	104 kHz / 16
	CPU		Exp-512	20.4 kHz / 2600
STM32F407VG Discovery @168MHz	DMA	Yes	AES-128	980 kHz / 32
			Exp-512	88 kHz / 4096
STM32L475 IoT Node @80MHz	DMA	Yes	AES-128	684 kHz / 64
			Exp-512	40 kHz / 4096

as fast as possible, to achieve a reasonably high sampling rate (100-1000kHz). For the two STM32 devices this could be achieved by different ADC-DMA setups. For the ESP32-devkitC we had to use the ULP-CPU to sample the ADC fast, and a normal CPU task to sample it slow. That is because the ULP has only access to limited memory below 2048 bytes in total, but also has a dedicated ADC instruction for fast sampling. The finally-used sampling rates depending on the used board and algorithm are shown in Table 5.3. The shown values were estimated by measuring on an external pin, since it was not always clear from the software to find out the actual sampling rate. It might be feasible to achieve higher sampling rates on the ESP32 with its i2s-module.

5.3.2.4. mbedTLS implementation details

This subsection clarifies the implementation details of RSA and AES in the mbedTLS library, which we verified to be unchanged in the relevant parts for this section, for both library versions (2.6.1 and 2.12.0).

The RSA implementation in mbedTLS is based on CRT (which can be disabled) and can use the exponent blinding side-channel countermeasure if a sufficient source of randomness is provided to the library. The RSA private key function of mbedTLS (`mbedtls_rsa_private`) uses bigint arithmetic, and among other functions calls `mbedtls_ mpi_exp_mod`, which performs a sliding window modular exponentiation. That function is where we perform some of our leakage assessments on, with a 512-bit exponent and modulus. We use the message transmitted by UART as the *base* of this exponentiation. The other function we analyze is from the AES algorithm.

The mbedTLS library implements the AES using a T-table lookup based approach. Originally, the AES is a round-based block cipher, where four different operations *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey* are repeatedly applied on a 128 bit data block. A popular optimization is to implement those operations as a combination of multiple table lookups and a subsequent *XOR* operation. This optimization requires the precomputed T-tables, which take an input byte and output a 32 bit word. Besides the addition of the first round key and the last round, the remaining rounds are executed in pairs of two inside each loop iteration. Apart from these optimizations, the mbedTLS AES implementation does not diverge from the textbook AES encryption algorithm and does not include any side-channel countermeasures in particular.

5.3.3. Results

Before doing leakage assessment or CPA, we first show that ADC noise correlates with the power consumption of the board. Subsequently, leakage assessment is performed on AES and modular exponentiation of the mbedTLS library.

In all of the tested platforms, side-channel leakage was found in at least one of the tested cryptographic algorithms and operation modes (configurations) of the ADC. In many setups, generic noise was observable on the ADC, even when it was pulled to Vdd or GND. Just in a few setups, we actually observe zero variance in the ADC output, such that information leakage is impossible.

5.3.3.1. Preliminary Comparison between Voltage and ADC Noise

For this experiment, we use the STM32F407VG Discovery #1, and run the CPU in high and low activity phases that should be easily distinguishable. In high activity phases, we perform floating point operations, whereas during the intermediate low activity phases, we issue *nop*-commands.

In Figure 5.22, we show the average of 1000 traces of the supply voltage V_{dd} and supply current I_{dd} measured with an oscilloscope. Concurrently, we show the acquired samples of 6-bit ADC values at maximum sampling rate using DMA mode. Both were recorded during the same activity phases of the CPU on the STM32F407VG Discovery board. The ADC is connected to a floating pin, and neither to GND or V_{dd}. The different phases of workload activity phases can easily be visually distinguished in both of the traces. Our activity pattern can be identified in the externally collected traces for both V_{dd} and I_{dd} in the timeframe from $0\mu s$ to about $160\mu s$, whereas the data transfer of ADC traces to a workstation occurs after $160\mu s$. Likewise, the ADC average values in the bottom diagram reflect the activity in a clearly distinguishable way, albeit not with linear correlation.

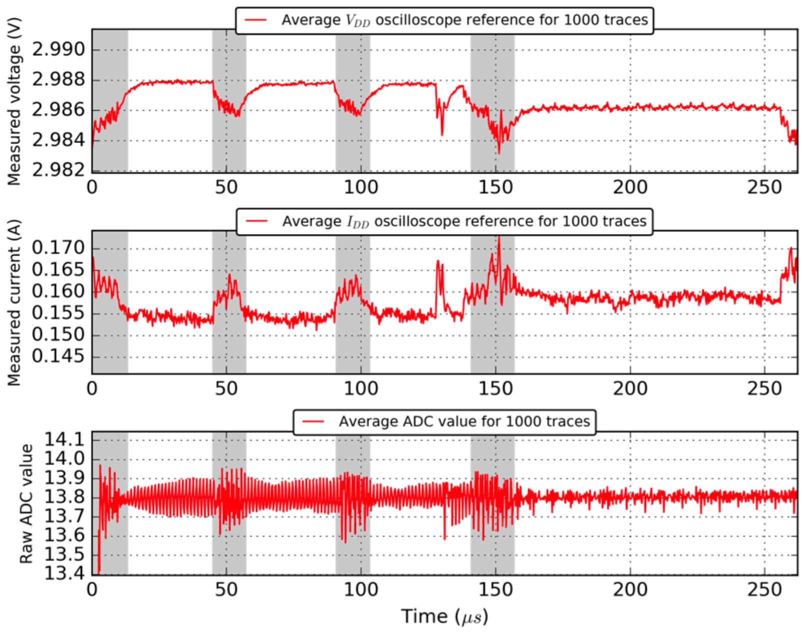


Figure 5.22.: Average over 1000 traces for oscilloscope measurements on V_{dd} and I_{dd} in the first two plots and average of concurrently measured ADC values when the ADC was set to 6-bit and the pin was not connected (N/C) on the bottom. High activity phases are marked grey.

5.3.3.2. Comparisons on Average ADC Traces of AES and Modular Exponentiation

The preliminary experiment to compare CPU activity phases already shows promising results. However, this experiment is a synthetic test case in which extremely high and low activity phases were chosen on purpose. Subsequently, we prove that even minor differences in the data processed by cryptographic algorithms affects the ADC noise in a systematic way. For that proof, we can already use the data required as a prerequisite for the leakage assessment we explained in Section 2.7. We collect two sets of traces while performing encryption operations. One of the sets contains the fixed traces, while the other set contains the random traces. For this example, we connect the pin used for ADC to GND and again use the same board as used in Section 5.3.3.1, STM32F407VG Discovery #1.

In Figure 5.23, we compare the ADC noise that occurs during sliding window exponentiation with a 512-bit secret exponent and modulus. In the first plot, an average over 100,000 (100k) ADC traces is shown, where an exponentiation is performed on the same fixed base. The second plot also shows the average, but with exponentiation on 100k different random bases. The red lines show the averages, while the grey background contains all the single traces. The differences between these plots are indeed distinguishable without further processing. Even more, a pattern is already visible in

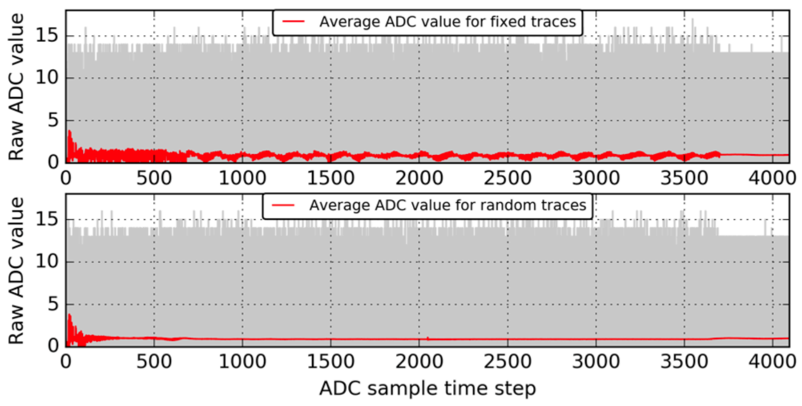


Figure 5.23.: Average over 100k traces for a fixed base exponentiated in a mbedTLS sliding window exponentiation, and 100k traces with each a random base exponentiated with the same secret exponent on the STM32F407VG Discovery #1. ADC connected to GND.

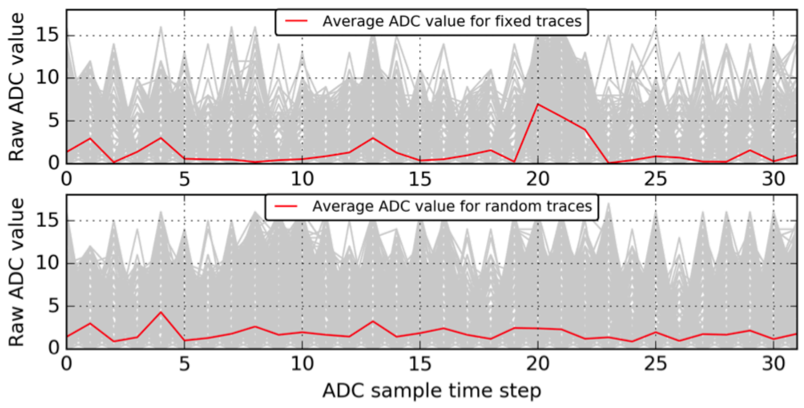


Figure 5.24.: Average over 1M traces for a fixed message encrypted with mbedTLS AES and the FIPS key, and 1M traces with each a random message encrypted with the same key on the STM32F407VG Discovery #1. ADC connected to GND.

the average of the fixed traces, which is smoothed out in the case of random traces. This example already shows that a power analysis attack for secret key extraction later on might be feasible.

Additionally to modular exponentiation (Mod-Exp), we also show an average of AES-128 for fixed and random messages in Figure 5.24. Since AES executes in much shorter time relative to the ADC speed, we can only acquire a few samples, which are at best 2-3 samples per AES round for this specific board. For AES, we collect 1,000,000 (1M) traces for each of the two sets of fixed and random messages. Similar to Mod-Exp, the differences between the traces are visible, with the most distinguishable sample point around sample 20.

5.3.3.3. Leakage Assessment on AES and Modular Exponentiation

We analyze the differences between the traces collected during fixed and random modular exponentiations and AES encryptions using the t-test methodology from [72, 73] as explained in Section 2.7. Exemplary, we show the results of leakage assessment on the ADC traces collected on the STM32F407VG Discovery, which we presented in the previous subsection. Further details from other platforms can be found in the appendix.

Figure 5.25 shows the first order leakage on the STM32F407VG Discovery for fixed-vs-random traces collected during modular exponentiation. The mathematical evaluation confirms the visual assessment from the previous section: The fixed and random traces are distinguishable, as the absolute $|t|$-value exceeds the threshold limit of 4.5 significantly.

Likewise, in Figure 5.26 we present the first order fixed-vs-random leakage during AES encryptions. Again, we confirm that the traces for fixed and random encrypted messages are clearly distinguishable, as the $|t|$-value is above the threshold.

Subsequently to the initial experiments, we perform an analysis on the boards introduced in the experimental setup, Section 5.3.2. First, we look into the ADC-observable leakages from AES, and secondly into modular exponentiation. This way, we also test the ADC noise characteristics at different operating frequencies. As we presented in the experimental setup in Table 5.4, for AES we had to use rather high sampling frequencies of the ADC on all the platforms ($104 - 980$kHz). For the ESP32-devkitC, no

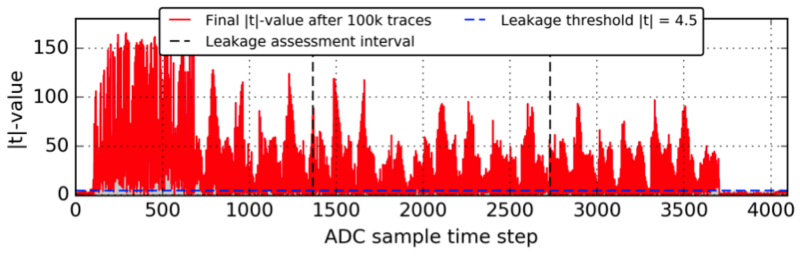


Figure 5.25.: First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #1.

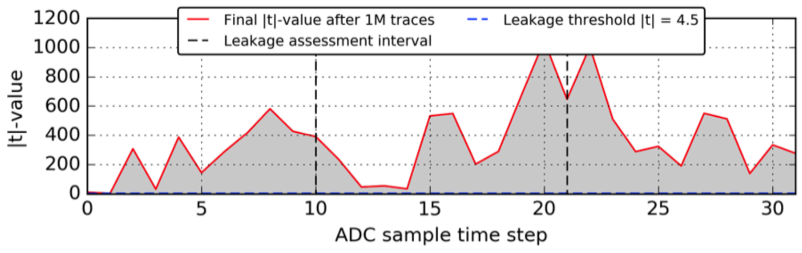


Figure 5.26.: First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery#1.

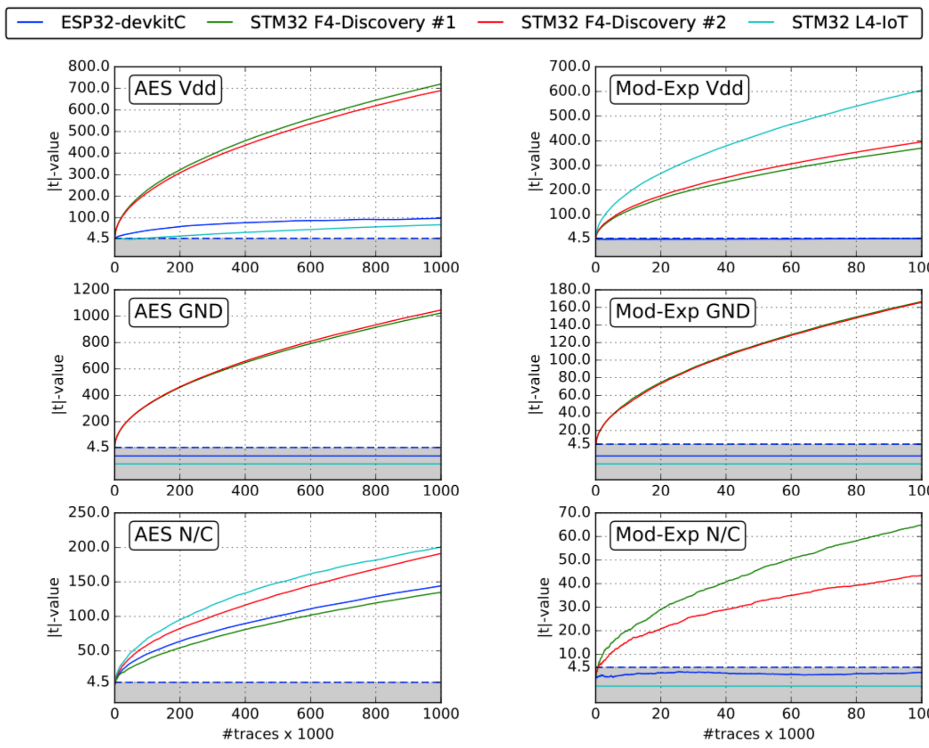


Figure 5.27.: Leakage Assessment on mbedTLS AES and modular exponentiation (Mod-Exp) with {Vdd, GND, N/C} connected to the ADC on all platforms. Flat lines on the bottom are constant zero, shifted for visibility.

higher frequency than 104kHz is reachable, which leads to less than 16 samples over the complete AES runtime. For modular exponentiation, the runtime is longer. Thus, we sample slower ($20-88$kHz) to collect only as much samples as the memory capacity of the internal SRAM.

In Figure 5.27, we show six plots of first order leakage assessments on AES and modular exponentiation (marked as 'Mod-Exp'), separated by algorithm and the connection of the ADC being Vdd, GND, or N/C, respectively. In these plots, the change of the highest $|t|$-value inside the leakage assessment interval (cf. Figure 5.25,Figure 5.26) is shown, over an increasing amount of traces used for the evaluation.

We start with the AES algorithm in the left column of Figure 5.27. We compare the platforms when the ADC is connected to Vdd. In this configuration, $|t|$ reaches values clearly beyond the confidence threshold of 4.5, suggesting that all platforms are leaking the information processed in the AES algorithm. For the case of a connection to GND, both samples of the STM32F407VG Discovery leak, but not the other boards. For the other boards, the ADCs actually output a constant value, such that Ground-Noise does not occur. In the case, when we have no connection (N/C) on the ADC, i.e. the pin is in a so-called floating state, all of the boards exhibit leakage ($|t| \gg 4.5$). We also looked into second order leakage. However, there were no changes with respect to the

Table 5.4.: Overview of Leakage Assessment over all tested Platforms and Configurations; ADC connected to {Vdd, GND, not connected (N/C)}. Amount of collected traces are 100k for modular exponentiation, and 1M for AES. The ADC was noise-free when σ=0.

Platform	**Leakage detected? ($t > 4.5$)**					
	AES-128 (ADC fast)			Mod-Exp-512 (ADC slow)		
	Vdd	GND	N/C	Vdd	GND	N/C
ESP32-devkitC @80MHz	yes	σ=0	yes	no[1]	σ=0	no[1]
STM32L475 IoT Node @80MHz	yes	σ=0	yes	yes	σ=0	σ=0
STM32F407VG Discovery #1 @168MHz	yes	yes	yes	yes	yes	yes
STM32F407VG Discovery #2 @168MHz	yes	yes	yes	yes	yes	yes

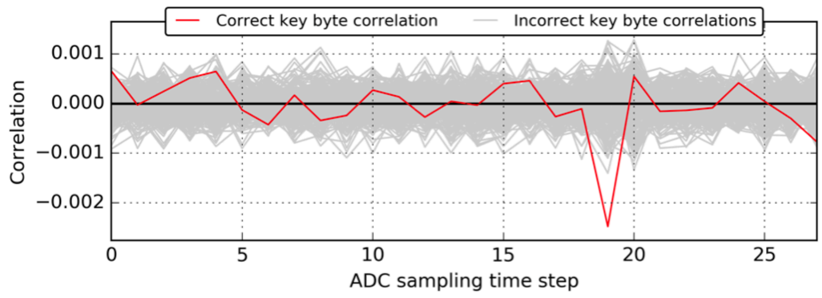
[1] For the center 1/3 trace. For the beginning and/or end of the cryptographic function, |t| was above 4.5. For more details check Section B.1, Figure B.2.

conclusion, if leakage is observed ($|t| > 4.5$), or not ($|t| \leq 4.5$).

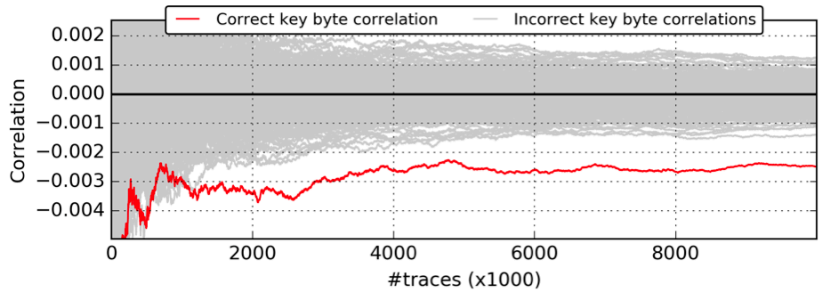
In the case of Mod-Exp, we only show first order leakage again in Figure 5.27, since there was no difference in the second order, similar to AES. We still compare the platforms with ADC connections to {Vdd, GND, N/C}. For Mod-Exp, again both STM32F407VG Discovery #1 and #2 leak in all of the tested cases. The other boards leak less than for AES. While the ESP32-devkitC t-value reaches beyond $|t| > 4.5$ for Vdd and N/C, it only does during the beginning or end phase of the modular exponentiation. We thus conclude negatively for the leakage assessment. This result might be due to an input value being copied or recoded inside the function. Interested readers can check the appendix, Figure B.2, for this detail. For the STM32L475 IoT Node, the ADC was sufficiently noisy at faster sampling rates for AES, except when the ADC was connected to GND. However, for modular exponentiation it also became entirely noise free when being not connected (N/C).

5.3.3.4. Summary of Leakage Assessments

In summary, the ADC settings such as the sampling frequency and connection to Vdd, GND or N/C levels affect the observable side-channel leakage. This relation exists, because the sampling frequency also changes the inherent noise characteristics. The connection of the ADC affects how it is coupled to the remaining parts of the system, particularly the digital (CPU and memory) subsystem. In most cases when the ADC shows any noise (sample data with $\sigma > 0$), the t-test detects leakage in the ADC data. In other cases, when the ADC is completely noise free ($\sigma = 0$), surely no information leakage is possible. The two boards that we used of the same type lead to almost the same results, leading to the conclusion that sample variation only has a minor effect. We summarize these results in Table 5.4. For reference, Section B.1 shows all leakage assessments across all the boards in detail, including the assessment over the complete time window when ADC samples were acquired (cf. Section 5.3.2.2).



(a) Total correlation after 10M traces for all 256 key byte candidates; The correlation with the correct key byte is marked red



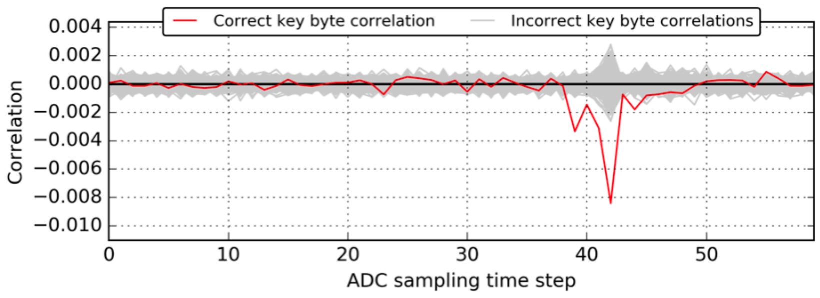
(b) Correlation progress over 10M traces for all 256 key byte candidates; The correlation with the correct key byte is marked red

Figure 5.28.: Results of a CPA attack on the 6th byte of the last secret round key of AES on the STM32F407VG Discovery #1 @168MHz with the ADC connected to GND and the program compiled with the -Os optimization option (like for leakage assessment).

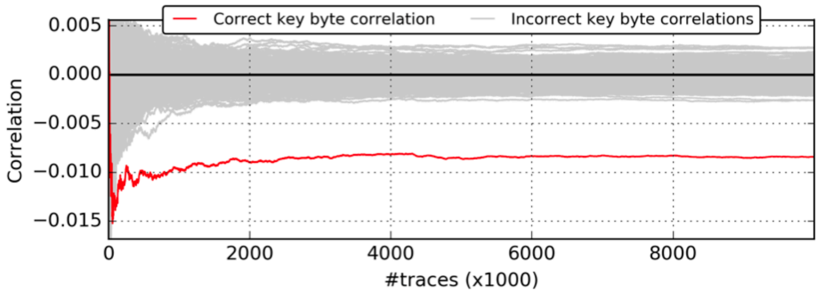
5.3.3.5. Correlation Power Analysis Attack on AES

In this subsection, we present results of CPA attacks on secret AES round keys in different setups. We perform a ciphertext-based CPA on the last round of AES over 10M ADC traces and show both the final correlations for each key byte candidate with the entire set of traces as well as the correlation progress over the amount of traces. We evaluate the different preprocessing and power model variants, which are explained in Section 2.8. Pre-aligning traces with a shift of ± 2 using normalized cross-correlation, and performing CPA with the standard S-box Hamming distance model is the most successful variant. The CPA experiments are performed on the STM32F407VG Discovery, which shows the most promising results during leakage assessment. Although we attacked all 16 bytes of the AES round key, only the best results for the respective setup are presented here. We state the total amount of recovered bytes for each setup and display the correlation plots for all key bytes in the appendix.

We first evaluate a CPA attack on 10M traces using the same parameters as for leakage assessment, when the ADC pin is connected to GND. Here, AES takes about $13\mu s$, leading to an estimated 17 samples during the complete AES function call to



(a) Total correlation after 10M traces for all 256 key byte candidates; The correlation with the correct key byte is marked red



(b) Correlation progress over 10M traces for all 256 key byte candidates; The correlation with the correct key byte is marked red

Figure 5.29.: Results of a CPA attack on the 12th byte of the last secret round key of AES on the STM32F407VG Discovery #2 @56MHz with the ADC connected to Vdd and the program compiled with the -O0 optimization option.

`mbedtls_internal_ aes_encrypt(..)`. In this setup, we are able to recover 2 secret key bytes in total, where the best correlation is seen for the 6th byte. In Figure 5.28a, the total correlation with all of the 256 possible values of the 6th byte of the last round key after 10M ADC traces is shown. A small peak in the last part of the correlation values indicates the sampling point that corresponds to the last AES round operation as well as the correct value of the last round key byte, which has the maximum correlation value at that point. However, the differences in correlation between the correct and the incorrect key bytes and the peak in general are very small.

In addition to evaluating CPA attacks on the previous setup for leakage assessment, we also adapt the compilation and frequency parameters to achieve a higher relative ADC sampling rate. Additionally we found to be more successful when the ADC pin is connected to Vdd. In that case, the MCU is running at 56MHz and the program is compiled with -O0 optimization, making AES take about 40μs to compute. Due to the on-chip clock network, the ADC is running slower, but faster relative to AES. This relative improvement leads to about 43 ADC samples for the entire AES encryption. Furthermore, we activate the DMA-based ADC sampling from the encryption task directly. This avoids any misalignment and synchronizes exactly with the beginning of the AES computations. In this setup, 6 secret key bytes can be recovered and the best correlation appears when attacking the 12th byte. The results can be seen in Figure 5.29. A peak during the last part of the encryption is clearly visible, again indicating the last AES round. Furthermore, we see that the correct key byte, which is marked red, correlates much clearer with the collected traces than the incorrect key bytes.

We conclude that key recovery attacks on the AES with data from ADC traces are generally possible, albeit the success depends on the overall system parameters, such as the clock speed, the ADC pin connection, and possibly even the code optimization at compile time. For data collected with the ADC pin left unconnected, we were unable to recover secret key bytes successfully.

5.3.4. Discussion

Our results prove the existence of a correlation between the data processed in a microcontroller, and the noise that can be observed in their integrated ADCs. The correlation is strong enough to distinguish the data processed in cryptographic algorithms, running on a CPU in the system. By leakage assessment it was shown that this observation is valid in most cases. Furthermore, we proved that the leakage can be sufficient to perform a CPA-based key recovery attack on AES. Due to the protection mechanism that can be enabled for the full RSA implementation of the used mbedTLS library, we assume that more advanced attacks are required for that, but are generally feasible [158–160].

The performed experiments reveal an underlying problem of highly integrated mixed-signal systems that consist of analog and digital components in a single chip. Such a level of SoC integration is an increasing trend in many hardware platforms for IoT applications and beyond. In these systems, both power supply based coupling effects as well as crosstalk can cause the noise in the analog part, which can then be exploited by anyone with access to the data measured in the analog circuit.

5.3.4.1. Practical relation to the adversarial model

Applications based on microcontrollers or SoCs use the ADC for various tasks, such as audio streaming or other sensor measurements. For power saving applications, the respective ADC pin is often pulled to GND, the supply voltage (V_{dd}) or left unconnected. Any of these ADC pin configurations can be the recommended way to save power, depending on the manufacturer and device. Often, it is possible to define by software to which pin the ADC is connected, or even internally connect it to GND or Vdd, such as Vref for the tested ESP32 chip. Some of these IoT or mobile applications make their data publicly available, or execute untrusted code in a memory protected area. In both ways, an adversary with the right access can get the necessary side-channel data to perform an attack.

Accidentally allowing a high sampling rate in publicly accessible IoT sensors can already be an attack vector from which sufficient leakage can be collected. For instance, a reference application for one of the tested platforms contains a web server which accepts any ADC sampling rate from its website. Any website hosted on that web server can control the sampling rate through JavaScript. JavaScript in turn can be manipulated by any user that accesses the website. In effect, if this reference application is used as the basis of a product, it introduces a threat. On the other hand, 3rd party smartphone applications often need to use various analog sensors of the system, which could contain the required side-channel information to perform an attack. Even a typical audio sampling rate is already fast enough to distinguish differences in modular exponentiation on the tested platforms, and it was already reported that such a low sampling rate can be sufficient to attack RSA [161].

Our experimental setups were chosen for comparability across different vendor systems, and additionally to allow methods for leakage assessment to be performed easily. Yet, these setups are sufficient to prove that ADC noise is generally a possible source of side-channel information. Using the aforementioned example, this could actually be exploitable in some existing products, and adds a dangerous new way to acquire power side-channel information completely remotely.

If an attack on a real system can succeed depends on additional aspects. Next to the basic requirement of access to ADC data, it is also required that the data can be sampled during cryptographic operations, and the collected traces to be aligned properly. However, it was shown that even in full commercial implementations, many of such obstacles can still be circumvented, and complex attacks can be performed by considering more aspects of a full system [11, 137, 162, 163]. For instance, it is often still possible to find a way of synchronization. At least an estimation at which time an encryption starts can often be achieved through the behavior of the overall application. A remote user can thus estimate which part of sensor data might contain exploitable side-channel leakage.

Another aspect in real systems is the side-channel data being modulated on top of other sensor measurement data, or the available sampling rate. However, this should only increase the number of traces required for differential attacks, which are specifically suited to cope with such situations. Often the sensor data is not much of a problem if it changes rather slowly, for instance a temperature sensor. It was also shown that a sampling rate below the expected side-channel leakage can still be sufficient to perform a full attack [157, 161].

Our results imply that even sensor data that a microcontroller measures and sends over a digital connection made available online can leak sensitive information which is accessible from anywhere in the world. In many scenarios, leaking the secret keys of a sensor node might be just a small issue, but in other scenarios a more sophisticated attacker could use such information as part of a larger scale attack, for instance on SCADA systems [164, 165].

5.3.4.2. Mitigating the threat

As our results suggest, it is of high importance that designers of embedded systems or integrated circuits for critical applications are aware of a potential security vulnerability through integrated analog components. That fact is the most important take-away message from the investigations performed in this work. We believe that noise of analog components should not be just characterized as a noise margin, but needs to be assessed for information leakage from security-related digital components in the system.

When designers are aware of this possible threat, it is achievable to design proper mitigations, depending on the application. The task of mitigation, for instance, can be considered in the hardware design of the ADC module to reduce or completely remove the noise-related leakage. In small IoT systems in which all executed code can be trusted, information leakage and possible attacks require a certain level of care, and we suggest either of the following options as a possible isolation practice:

- It must be guaranteed that any analog measurements can only take place mutually exclusive to security-related computations.
- If exclusive measurements cannot be guaranteed, a leakage assessment needs to be performed on all analog measurement data that is made accessible to possible attackers. If the data contains leakage, it has to be handled with the same security level as the secret data that is processed in the system.
- Filter the ADC data in a way that leakage cannot be assessed anymore. For instance, a filter for noise or specific frequencies might make attacks infeasible, or reduce effective sampling rates.

However, in multi-user systems in which memory protection is used among unprivileged tasks that are not fully trusted, more serious care has to be taken in order to prevent one of the tasks from extracting the secrets of another task. For instance, underprivileged tasks should not be allowed to get arbitrary ADC measurements. This could mean that in systems like a smartphone, a task that records the microphone on one of the ADCs could potentially run power analysis side-channel attacks on the remaining system. More research is needed to evaluate how widespread this threat is in existing systems.

5.3.5. Section conclusion

Mixed-signal systems such as microcontrollers and other SoCs are increasingly adopted in Internet-of-Things and personal computing appliances. In these systems, digital logic such as a CPU causes noise in the analog components, for instance in the ADCs used for a microphone, or any environmental sensors. In this section, leakage assessment was performed on the noise of ADC data for various platforms. In the majority of cases, leakage was detected. In one case we demonstrated a successful key recovery attack on AES proving that the leakage can be exploited. Previous works have explored a similar software-based power analysis side-channel, and this section generalizes those results. It is now confirmed that power analysis side-channels are not just a threat for attackers with physical access to the device. Any sensory data that leaves a system might contain enough correlated noise that could be exploited to perform power analysis attacks. Thus, we want to stress the importance to either use better isolation between analog and digital subsystems, or protect cryptographic implementations against power analysis attacks, even if local attackers are not considered in the threat model.

Part III.

Related Work and Summary

6. Related Work and Countermeasures

The sections in this chapter were partially overtaken from previously published works included in this thesis, which were co-authored with (in no particular order): Falk Schellenberg, Jonas Krautter, Fabian Oboril, Saman Kiamehr, Amir Moradi and Mehdi B. Tahoori.

Next to the work carried out in this thesis, fault or side-channel attacks that can be exploited even remotely through software are recently increasing. Probably the most impactful seminal findings have been the Spectre and Meltdown speculative execution attacks that use cache timing side-channels [166, 167] as covert information channels [11, 12] to exfiltrate information from a speculative execution context. These impact a wide range of mainstream CPUs in use, and have thus motivated further research in the direction of *microarchitectural attacks*. However, in contrast to that, this thesis focuses on a lower level, in which the electrical characteristics of the system can be exploited and impacted from the software side. Thus, specifically power analysis and timing or voltage-based fault attacks are the main focus.

6.1. Voltage- and Timing-based Fault Attacks through Software access

The earliest shown voltage-based attack that can be influenced through user software has been *rowhammer* [95], which demonstrates that faults can be caused in DRAM through malicious access patterns on the memory. These access patterns use repetitive reading of the same memory line, which can cause faults in an adjacent line, leading to bit flips. Because that adjacent memory line might be an important part of protected operating system memory, certain bit flips can eventually lead to privilege escalation, and finally exfiltrate secret information [95, 168–170]. This attack has been first shown by programs running on the system, but later it has even been demonstrated through JavaScript in a web browser [96], or for privilege escalation in virtualized environments [171, 172]. Furthermore, truly remote settings that only need a network connection have also been explored [173–175].

Another example in which faults can be caused in software is an ARM-based SoC, secured by *ARM TrustZone*. TrustZone is designed to securely isolate various components of an SoC, and can also establish a *Trusted Execution Environment* (TEE) for trusted applications on the ARM cores. The *CLKSCREW* [137] attack shows how power management can be reprogrammed and exploited even by untrusted applications on the ARM CPU, affecting trusted applications or hardware in the SoC which otherwise should not be accessible. Using CLKSCREW, the used RSA signature scheme in TrustZone can also be subverted, leading to execution of self-signed applications. More recently, we also saw control over power management being used to attack Intel SGX in [176–178].

When looking into FPGAs, this thesis shows the first results on injecting transient faults merely through reconfiguration in Chapter 4. Others have extended these attacks. In [179], timing faults are caused that make a random number generator misbehave. Other works have shown alternate ways to cause voltage drops or inject timing faults. By specific memory access patterns to FPGA-internal BRAM, sufficient voltage drop to cause faults can be generated [180]. On the other hand, alternate ring oscillator designs, which do not require a combinational loop, have been shown in [109]. In [181], it is further characterized what exact on-chip spatial and timing conditions are needed to inject faults successfully.

6.2. Power Analysis Side-Channel Attacks through Software access

Before this thesis, preliminary studies have analyzed the impact of electrical coupling between logically isolated components in the same chip from a security perspective in [127, 128]. In one study, the dedicated internal power sensors of an FPGA were used, but were found to be insufficient for power analysis attacks [129]. In this work, we presented a power analysis attack inside FPGAs in Chapter 5.

Since the first publication included as Section 5.1 from 2018, others have also worked on this topic. Most of the works use the sensors introduced in Chapter 2, to either count oscillations of ROs, or use TDC sensors. By using the TDC sensors as presented in Chapter 2, a higher sampling rate can be achieved, reaching higher sampling rates than the oscillator-based sensor. In [105] RO sensors are used to attack implementations of a simple textbook RSA, both in FPGA fabric, as well as a co-residing CPU in an SoC integrating CPUs and FPGA part. That extends the results from Section 5.1 from FPGA to SoC level, while the results in Section 5.2 again lift this to PCB level. Some recent works have also looked into improving the methods of sensing or evaluating the data [182, 183], and attacking more than just textbook RSA in a SoC platform [184].

In 2018, it was shown how the noise in mixed-signal systems can be exploited by a semi-remote attack [147], i.e. from a short distance. In their work, small wireless IoT devices are attacked, which integrate the wireless radio circuit together with digital logic. When the digital logic performs cryptographic operations, leakage through the silicon substrate affects the analog radio circuit. If that circuit is used, a fraction of the side-channel leakage from the cryptographic operations can actually be observed in the electromagnetic wave emitted by the radio circuit into the nearby environment. In this way, EM leakage is essentially extended to a larger range than usual [157]. This leakage was sufficient to extract the secret keys of a simple *tinyAES* implementation from within a distance of up to 10 meters, and 1 meter for *mbedTLS*. A similar effect of leakage among chip components was shown for microcontrollers in [185], in which side-channel leakage could also be observed on digital port pins not connected to the power supply. Thus, it is also an indication that an ADC could observe such leakage if it is connected to a port pin from the inside. Extending the work presented in Section 5.3, O'Flynn et al. [186] uses the noise of an ADC to perform power analysis attacks targeting the secrets of an ARM TrustZone-M implementation.

6.3. Countermeasures

Both categories of attacks — fault or side-channel — require pragmatic solutions. For FPGAs, some initial solutions have already been published in collaboration with the author of this thesis, while minor suggestions have already been given in the respective earlier chapters. In general, power analysis side-channel attacks could be tackled with traditional countermeasures, such as hiding and masking schemes. However, given the specific nature of FPGA on-chip attacks, other types of countermeasures can also be developed.

One of these countermeasures is *Active Fencing*, which takes chip-wide floorplanning and spatial relations of attacker and victim circuits into consideration [187]. Earlier isolation techniques for FPGA security already suggest to put unused logic slices between victim and attacker circuits inside the FPGA [98], which should be able to protected against attacks that are based on crosstalk between adjacent wires of victim and attacker. Active fencing goes one step further, and instead puts active logic in those slices. That logic can cancel out a significant degree of side-channel leakage that travels through the on-chip PDN. In [187] it is demonstrated that a CPA attack can require 166 $\times$ more traces when *Active Fencing* is applied.

Another possible countermeasure is to perform a check on the bitstream, which was briefly mentioned for DoS in Section 4.1. A supervisor is integrated that reverse engineers and checks bitstreams before they are loaded into the FPGA as a sort of *FPGA Anti-Virus*. By analyzing the bitstreams for malicious signatures, circuit configurations used for attacks can be detected and prevented to execute on the FPGA. With perfect detection accuracy, all types of fault and side-channel attacks presented here could be prevented. Like all detection approaches, the practical difficulties are in suggesting signatures that allow to detect all malicious bitstreams, without rejecting actually benign bitstreams. In [188] and [189] various signatures and detection methods are suggested. Fundamental circuit properties are formulated that can be used for indirect sensing of voltage fluctuations or to cause faults through voltage drop, as was presented in the attacks in Chapter 4 and Chapter 5. In [189], these properties are then evaluated on a broad range of benchmark circuits.

6.4. Cross-coupling based Side-Channel Analysis Attacks in FPGAs

In 2007, Huffmire et al. [98] discussed potential countermeasures against security threats in an FPGA, in which various IP cores with unknown trust are integrated together. One of the discussed threats is a potential physical influence when wires from different designs are routed through the same switch matrix. As an isolation mechanism, so-called *moats* provide what is considered as *physical isolation* by adding unused FPGA slices between logical blocks, while drawbridges are used for a more restricted communication [98]. Major FPGA vendors such as Xilinx and Intel (formerly Altera) picked this up to suggest similar design flows for secure isolation [124, 190]. This countermeasure is considered a physical separation, however it misses the shared PDN, and by that can not prevent the attacks shown in Chapter 4 and Chapter 5.

However, these isolation mechanisms protect against another type of attack that has just recently been shown. In [125] it was shown first that long wires inside a single switch matrix can influence each other. Later on, it was shown that this is sufficient for a side-channel attack to extract information in [153]. Thus some studies have followed to analyze the exact behavior of this long wire coupling effects [191, 192] Since moats put additional space, that cross influence can be prevented, at the cost of unused slices.

6.5. Results of this Thesis and similar Related Work

Here we summarize the experiments that were shown in previous chapters, and also some additional results, showing how various FPGA platforms from different vendors are vulnerable to these attacks. Typically one type of attack was feasible on any of the tested platforms. This overview is provided in Table 6.1, also including some of the related work.

Table 6.1.: Overview of experimental results on side-channel or fault attacks in FPGAs, systems containing FPGAs, or FPGA-based SoCs.

Board	Attack successful?		
	Voltage Drop-based Denial of Service	**Voltage Drop-based Timing Fault Injection**	**Key Recovery by Side-Channel**
Intel Terasic DE0-Nano-SoC	–	Yes, this work	–
Intel Terasic DE1-SoC	Yes, this work	Yes, this work	–
Intel Terasic DE2-115	–	–	Yes, [153][1]
Intel Terasic DE4	–	Yes, this work	–
Lattice ECP5 5G Evaluation Board	–	–	Yes, this work
Lattice iCE40-HX8K Breakout Board	Yes, this work	Yes, this work	Yes, this work
Xilinx Artix-7 Basys-3	–	–	Yes, this work
Xilinx Kintex-7 KC705	Yes, this work	– [5]	–
Xilinx Pynq Zynq-ZC7020	Yes, this work[2]	– [5]	Yes, this work
Xilinx Spartan-6 SAKURA-G	–	–	Yes, this work[4]
Xilinx Ultrascale VCU108	Yes, this work	Yes, this work	–
Xilinx Virtex-6 ML605	Yes, this work	– [5]	–
Xilinx Virtex-7 VC707	–	Yes, [179]	–
Xilinx Zedboard Zynq-ZC7020	Yes, this work[2]	– [5]	Yes, [105][3]

[1] Information leaks through cross-coupling of adjacent wires. This is less of a threat, since coupling is prevented if interconnect matrices are not shared between multiple designs [153], which is also recommended by standard secure design flow practices [98, 124].

[2] It affects the whole SoC including the integrated ARM Cortex-A9 Dual-Core.

[3] Sufficient leakage for key recovery was also shown from CPU to FPGA in the same SoC

[4] In [23] it was additionally shown to work from one FPGA in the system (connected to the same power supply) to another, on board-level.

[5] A simple experiment was conducted, but the devices crashed before timing violations occured – it might still be possible with more effort.

7. Conclusion and Perspectives

7.1. Conclusion

The experiments conducted for this thesis show that new security threats emerge with new use-cases of FPGA hardware, such as multi-tenancy, or as accelerators in a multi-user system. Albeit FPGAs are intended as digital circuits, the methods used in this thesis allow that voltage fluctuations can be generated or observed inside FPGA designs, in a way sufficient to perform attacks. Thus, with software-based reconfiguration of an FPGA, physical attacks such as power analysis and fault attacks are feasible inside the FPGA, but also to other chips that share the same power supply. Previously, such attacks were believed to require local access to the device under attack, and dedicated test and measurement equipment. By this thesis these threats are now lifted to a potentially remote attacker, exemplifying that they should not be disregarded in threat modeling.

The results for FPGAs also indicate that the electrical level of semiconductor chips in general should not be disregarded for a system-wide security analysis. Thus, to demonstrate this generality, the thesis also analyzes a similar threat for mixed-signal IoT devices. In that case, it reveals that power side-channel information is indirectly available as the noisy part of on-chip ADCs.

In conclusion, this thesis has shown that the electrical level of semiconductor chips, foremost FPGAs, is a feasible attack vector, even when the attacker has no direct physical access to the device. The results of this thesis suggest that solutions have to be found for multi-tenant FPGA security, before they can be used responsibly on a wider scale. Considering other semiconductor devices, system-wide security threat modeling needs to include the electrical level, and not only when a local attacker is considered.

7.2. Perspectives on Applications, Research, and Education

In the future, we need to consider multi-tenant FPGA systems, or any other system with a shared power supply to be at potential risk of an attack, if on-chip resources can be misused to either measure or manipulate power supply voltage. Thus, countermeasures for at least the threats shown in Chapter 4 and Chapter 5 need to be found. Specifically for these systems, we briefly discussed common recommendations or countermeasures in the respective chapters, and also in the related work, Chapter 6. These countermeasures so far try to either use existing FPGA fabric, or a more restrictive system-level approach in which an FPGA design can only be loaded after it gets checked.

More generally, these countermeasures are very specific to the application so far, and do not yet solve or grasp the underlying problem of sharing a power distribution network among components with different security privileges. For that, the underlying electrical level needs to be more carefully analyzed in each new chip generation and stands as an open research problem, which has also been introduced into the research community recently.

Finally, as an interesting byproduct, the results of this thesis are also valuable for education. Since this thesis showed that low-cost FPGA boards alone are sufficient to perform practical fault or side-channel attacks, expensive test and measurement equipment is not required anymore. By that, each student can be provided with their own board in which all necessary experiments can be performed, even at home. Together with an accompanying lecture, a course *Practical Introduction into Hardware Security* is already established at the Karlsruhe Institute of Technology, in which we use these benefits to teach fault and power analysis attacks.

Part IV.

Appendix

Bibliography

[1] Y. Xu, O. Muller, P.-H. Horrein, and F. Petrot, “HCM: An abstraction layer for seamless programming of DPR FPGA,” in *22nd International Conference on Field Programmable Logic and Applications (FPL).* IEEE, aug 2012.

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

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

[4] S. A. Fahmy, K. Vipin, and S. Shreejith, “Virtualized FPGA Accelerators for Efficient Cloud Computing,” in *CloudCom.* IEEE Computer Society, 2015, pp. 430–435.

[5] S. Yazdanshenas and V. Betz, “Interconnect Solutions for Virtualized Field-Programmable Gate Arrays,” *IEEE Access*, vol. 6, pp. 10 497–10 507, 2018.

[6] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, “A Reconfigurable Fabric for Accelerating Large-scale Datacenter Services,” in *International Symposium on Computer Architecuture (ISCA)*, ser. ISCA ’14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 13–24. [Online]. Available: http://dl.acm.org/citation.cfm?id=2665671.2665678

[7] Amazon EC2 F1 Instances. [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/

[8] I. Corporation. (2017) Intel FPGAs Power Acceleration-as-a-Service for Alibaba Cloud | Intel Newsroom. [Online]. Available: https://newsroom.intel.com/news/intel-fpgas-power-acceleration-as-a-service-alibaba-cloud

[9] H. Cloud. (2020) FPGA Accelerated Cloud Server (FACS). [Online]. Available: https://www.huaweicloud.com/en-us/product/fcs.html

[10] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O’Flynn, “IoT Goes Nuclear: Creating a ZigBee Chain Reaction,” in *IEEE Symposium on Security and Privacy (S&P).* IEEE, May 2017, pp. 195–212.

[11] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre Attacks: Exploiting Speculative Execution," *ArXiv e-prints*, Jan. 2018.

[12] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown," *ArXiv e-prints*, Jan. 2018.

[13] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter Variations and Impact on Circuits and Microarchitecture," in *Proceedings of the Design Automation Conference (DAC)*, ser. DAC. New York, NY, USA: ACM, 2003, pp. 338–342. [Online]. Available: http://doi.acm.org/10.1145/775832.775920

[14] M. S. Gupta, J. A. Rivers, P. Bose, G.-Y. Wei, and D. Brooks, "Tribeca: Design for PVT Variations with Local Recovery and Fine-grained Adaptation," in *International Symposium on Microarchitecture (MICRO)*, ser. MICRO 42. New York, NY, USA: ACM, 2009, pp. 435–446. [Online]. Available: http://doi.acm.org/10.1145/1669112.1669168

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

[16] S. Das, P. Whatmough, and D. Bull, "Modeling and Characterization of the System-Level Power Delivery Network for a Dual-Core ARM Cortex-A57 Cluster in 28nm CMOS," in *International Symposium on Low Power Electronics and Design (ISLPED)*, July 2015, pp. 146–151.

[17] R. Zhang, K. Wang, B. H. Meyer, M. R. Stan, and K. Skadron, "Architecture implications of pads as a scarce resource," in *International Symposium on Computer Architecture (ISCA)*, ser. ISCA '14. Piscataway, NJ, USA: ACM/IEEE, 2014, pp. 373–384.

[18] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "Analysis of Transient Voltage Fluctuations in FPGAs," in *Proceedings of the International Conference on Field-Programmable Technology (FPT)*, China, Dec. 2016, pp. 12–19.

[19] ——, "An Experimental Evaluation and Analysis of Transient Voltage Fluctuations in FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 10, pp. 1817–1830, Oct. 2018.

[20] D. R. E. Gnad, F. Oboril, and M. B. Tahoori, "Voltage Drop-based Fault Attacks on FPGAs using Valid Bitstreams," in *International Conference on Field Programmable Logic and Applications (FPL)*, IEEE. IEEE, Sep. 2017, pp. 1–7.

[21] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, suitable for DFA on AES," *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, vol. 2018, no. 3, 2018.

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

[23] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "Remote Inter-Chip Power Analysis Side-Channel Attacks at Board-Level," in *International Conference on Computer-Aided Design (ICCAD)*, Nov 2018, pp. 1–7.

[24] D. R. E. Gnad, J. Krautter, and M. B. Tahoori, "Leaky Noise: New Side-Channel Attack Vectors in Mixed-Signal IoT Devices," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 3, pp. 305–339, May 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8297

[25] D. R. E. Gnad, F. Schellenberg, J. Krautter, A. Moradi, and M. B. Tahoori, "Remote Electrical-level Security Threats to Multi-Tenant FPGAs," *IEEE Design Test*, 2020.

[26] P. Larsson, "Power supply noise in future IC's: a crystal ball reading," in *IEEE Custom Integrated Circuits Conference (CICC)*, 1999, pp. 467–474.

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

[28] K. Arabi, R. Saleh, and X. Meng, "Power Supply Noise in SoCs: Metrics, Management, and Measurement," *IEEE Des. Test. Comput.*, vol. 24, no. 3, pp. 236–244, 2007.

[29] D. Klokotov, J. Shi, and Y. Wang, "Distributed Modeling and Characterization of On-chip/System-level PDN and Jitter Impact," in *DesignCon*, 2014.

[30] V. J. Reddi, S. Kanev, W. Kim, S. Campanoni, M. D. Smith, G. Y. Wei, and D. Brooks, "Voltage Smoothing: Characterizing and Mitigating Voltage Noise in Production Processors via Software-Guided Thread Scheduling," in *International Symposium on Microarchitecture*, Dec 2010, pp. 77–88.

[31] A. Muhtaroglu, G. Taylor, and T. Rahal-Arabi, "On-die droop detector for analog sensing of power supply noise," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 4, pp. 651–660, April 2004.

[32] T. Wang, C. Zhang, J. Xiong, and Y. Shi, "On the Deployment of On-Chip Noise Sensors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 4, pp. 519–531, April 2014.

[33] E. Seevinck, F. J. List, and J. Lohstroh, "Static-noise margin analysis of MOS SRAM cells," *IEEE Journal of Solid-State Circuits*, vol. 22, no. 5, pp. 748–754, 1987.

[34] Henry J. Zhang, "Basic Concepts of Linear Regulator and Switching Mode Power Supplies," Analog Devices, 2013.

[35] A. Nalamalpu, N. Kurd, A. Deval, C. Mozak, J. Douglas, A. Khanna, F. Paillet, G. Schrom, and B. Phelps, “Broadwell: A family of IA 14nm processors,” in *Symposium on VLSI Circuits (VLSIC)*, June 2015, pp. C314–C315.

[36] “AN-711 LM78S40 Switching Voltage Regulator Applications,” Texas Instruments, 2004–2013.

[37] X. Aragones, J. L. Gonzalez, and A. Rubio, *Analysis and solutions for switching noise coupling in mixed-signal ICs.* Dordrecht: Springer Science & Business Media, 1999.

[38] D. K. Su, M. J. Loinaz, S. Masui, and B. A. Wooley, “Experimental results and modeling techniques for substrate noise in mixed-signal integrated circuits,” *IEEE Journal of Solid-State Circuits*, vol. 28, no. 4, pp. 420–430, April 1993.

[39] T. Stöhr, M. Alt, A. Hetzel, and J. Koehl, “Analysis, Reduction and Avoidance of Crosstalk on VLSI Chips,” in *International Symposium on Physical Design (ISPD).* New York, NY, USA: ACM, 1998, pp. 211–218.

[40] F. Fiori, “On the susceptibility of analog circuit to EMI,” in *Analog Circuit Design.* Springer, 2007, pp. 183–202.

[41] B. Le, T. W. Rondeau, J. L. H. Reed, and C. W. Bostian, “Analog-to-digital converters,” *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 69–77, Nov. 2005.

[42] National Instruments, “ADC Dynamic Characteristics Measurement Reference Design,” 2015, http://www.ni.com/example/8319/en/. [Online]. Available: http://www.ni.com/example/8319/en/

[43] S. Aouini and G. W. Roberts, “A Predictable Robust Fully Programmable Analog Gaussian Noise Source for Mixed-Signal/Digital ATE,” in *IEEE International Test Conference*, Oct. 2006, pp. 1–10.

[44] K. M. Zick, M. Srivastav, W. Zhang, and M. French, “Sensing Nanosecond-scale Voltage Attacks and Natural Transients in FPGAs,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays.* New York, NY, USA: ACM, 2013, pp. 101–104. [Online]. Available: http://doi.acm.org/10.1145/2435264.2435283

[45] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, “Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation,” in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 36. Washington, DC, USA: IEEE Computer Society, Dec 2003, pp. 7–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=956417.956571

[46] E. Stott, J. M. Levine, P. Y. K. Cheung, and N. Kapre, “Timing Fault Detection in FPGA-Based Circuits,” in *IEEE Field-Programmable Custom Computing Machines (FCCM)*, May 2014, pp. 96–99.

[47] A. Brant, A. Abdelhadi, D. H. H. Sim, S. L. Tang, M. X. Yue, and G. G. F. Lemieux, "Safe Overclocking of Tightly Coupled CGRAs and Processor Arrays using Razor," in *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, April 2013, pp. 37–44.

[48] K. Minkovich and J. Cong, "Mapping for Better Than Worst-case Delays in LUT-based FPGA Designs," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '08. New York, NY, USA: ACM, 2008, pp. 56–64. [Online]. Available: http://doi.acm.org/10.1145/1344671.1344681

[49] J. S. J. Wong, P. Sedcole, and P. Y. K. Cheung, "Self-Measurement of Combinatorial Circuit Delays in FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 2, pp. 10:1–10:22, Jun. 2009. [Online]. Available: http://doi.acm.org/10.1145/1534916.1534920

[50] K. M. Zick and J. P. Hayes, "Low-cost Sensing with Ring Oscillator Arrays for Healthier Reconfigurable Systems," *ACM Trans. Reconfigurable Technol. Syst. (TRETS)*, vol. 5, no. 1, pp. 1:1–1:26, Mar. 2012. [Online]. Available: http://doi.acm.org/10.1145/2133352.2133353

[51] A. L. Masle and W. Luk, "Detecting power attacks on reconfigurable hardware," in *International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2012, pp. 14–19.

[52] C. T. Gray, W. Liu, W. A. M. V. Noije, T. A. Hughes, and R. K. Cavin, "A sampling technique and its CMOS implementation with 1 Gb/s bandwidth and 25 ps resolution," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 3, pp. 340–349, Mar 1994.

[53] J. Kalisz, R. Szplet, J. Pasierbinski, and A. Poniecki, "Field-Programmable-Gate-Array-based Time-to-Digital converter with 200-ps Resolution," *IEEE Transactions on Instrumentation and Measurement*, vol. 46, no. 1, pp. 51–55, Feb 1997.

[54] A. Aloisio, P. Branchini, R. Cicalese, R. Giordano, V. Izzo, and S. Loffredo, "FPGA implementation of a high-resolution time-to-digital converter," in *IEEE Nuclear Science Symposium (NSS)*, vol. 1, Oct 2007, pp. 504–507.

[55] C. Favi and E. Charbon, "A 17Ps Time-to-digital Converter Implemented in 65Nm FPGA Technology," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '09. New York, NY, USA: ACM, 2009, pp. 113–120. [Online]. Available: http://doi.acm.org/10.1145/1508128.1508145

[56] A. Drake, R. Senger, H. Deogun, G. Carpenter, S. Ghiasi, T. Nguyen, N. James, M. Floyd, and V. Pokala, "A Distributed Critical-Path Timing Monitor for a 65nm High-Performance Microprocessor," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2007, pp. 398–399.

[57] K. A. Bowman, C. Tokunaga, J. W. Tschanz, A. Raychowdhury, M. M. Khellah, B. M. Geuskens, S. L. L. Lu, P. A. Aseron, T. Karnik, and V. K. De, "All-Digital Circuit-Level Dynamic Variation Monitor for Silicon Debug and Adaptive Clock Control," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 9, pp. 2017–2025, Sept 2011.

[58] C. Lefurgy, A. Drake, M. Floyd, M. Allen-Ware, B. Brock, J. Tierno, J. Carter, and R. Berry, "Active Guardband Management in Power7+ to Save Energy and Maintain Reliability," *IEEE Micro*, vol. 33, no. 4, pp. 35–45, July 2013.

[59] J. Wu, "Several Key Issues on Implementing Delay Line Based TDCs Using FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 57, no. 3, pp. 1543–1548, June 2010.

[60] S. B. Örs, E. Oswald, and B. Preneel, *Power-Analysis Attacks on an FPGA – First Experimental Results.* Berlin, Heidelberg: Springer, 2003, pp. 35–50. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45238-6_4

[61] M. Masoomi, M. Masoumi, and M. Ahmadian, "A practical differential power analysis attack against an FPGA implementation of AES cryptosystem," in *Information Society (i-Society)*, June 2010, pp. 308–312.

[62] T. Iakymchuk, M. Nikodem, and K. Kępa, "Temperature-based covert channel in FPGA systems," in *International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, June 2011, pp. 1–7.

[63] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *IEEE Computer*, vol. 43, no. 10, pp. 39–46, Oct 2010.

[64] R. S. Chakraborty, I. Saha, A. Palchaudhuri, and G. K. Naik, "Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream," *IEEE Design & Test (D&T)*, vol. 30, no. 2, pp. 45–54, April 2013.

[65] L. Sauvage, S. Guilley, and Y. Mathieu, "Electromagnetic Radiations of FPGAs: High Spatial Resolution Cartography and Attack on a Cryptographic Module," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 4:1–4:24, Mar. 2009. [Online]. Available: http://doi.acm.org/10.1145/1502781.1502785

[66] L. Zussa, J.-M. Dutertre, J. Clediere, and A. Tria, "Power supply glitch induced faults on FPGA: An in-depth analysis of the injection mechanism," in *International On-Line Testing Symposium (IOLTS).* IEEE, jul 2013, pp. 110–115.

[67] I. Hadžić, S. Udani, and J. M. Smith, "FPGA Viruses," in *International Conference on Field Programmable Logic and Applications (FPL)*, P. Lysaght, J. Irvine, and R. Hartenstein, Eds. Berlin, Heidelberg: Springer, 1999, pp. 291–300. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-48302-1_30

[68] K. Kepa, F. Morgan, K. Kosciuszkiewicz, and T. Surmacz, "SeReCon: A Secure Dynamic Partial Reconfiguration Controller," in *Symposium on VLSI*, April 2008, pp. 292–297.

[69] A. Moradi, A. Barenghi, T. Kasper, and C. Paar, "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*. New York, NY, USA: ACM, 2011, pp. 111–124. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046722

[70] A. Bogdanov, A. Moradi, and T. Yalçin, "Efficient and side-channel resistant authenticated encryption of FPGA bitstreams," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Dec 2012, pp. 1–6.

[71] A. Moradi, M. Kasper, and C. Paar, *Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, ch. Topics in Cryptology CT-RSA, pp. 1–18. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-27954-6_1

[72] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi *et al.*, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, vol. 7, 2011, pp. 115–136.

[73] T. Schneider and A. Moradi, "Leakage assessment methodology," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2015, pp. 495–513.

[74] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," *Cryptographic Hardware and Embedded Systems - CHES 2004*, pp. 16–29, 2004.

[75] G. Piret and J.-J. Quisquater, "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad," in *Cryptographic Hardware and Embedded Systems - CHES 2003*, C. D. Walter, Ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 77–88.

[76] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb 2007.

[77] I. Kontorovitch, V. Drabkin, C. Houghton, and J. Laurent, "Measurements of impedance, current and worst-case noise on chip power delivery system under operating conditions," *DesignCon 2005*, 2005.

[78] A. Amouri, F. Bruguier, S. Kiamehr, P. Benoit, L. Torres, and M. Tahoori, "Aging effects in FPGAs: an experimental analysis," in *International Conference on Field Programmable Logic and Applications (FPL)*, Sept 2014, pp. 1–4.

[79] K. Aygün, M. J. Hill, K. Eilert, K. Radhakrishnan, and A. Levin, "Power Delivery for High-Performance Microprocessors." *Intel Technology Journal*, vol. 9, no. 4, pp. 273 – 283, 2005. [Online]. Available: http://www.redi-bw.de/db/ebsco.php/search.ebscohost.com/login.aspx%3fdirect%3dtrue%26db%3dbuh%26AN%3d19399418%26site%3deds-live

[80] H. Shi, G. Liu, A. Liu, A. Pannikkat, K. S. Ng, and Y. H. Yew, “Simultaneous switching noise in FPGA and structure ASIC devices, methodologies for analysis, modeling, and validation,” in *IEEE Electronic Components and Technology Conference (ECTC)*, 2006, pp. 8 pp.–.

[81] Z. Liu, S. Sun, and P. Boyle, “FPGA core PDN design optimization,” in *IEEE International Symposium on Electromagnetic Compatibility (EMC)*, Aug 2011, pp. 411–416.

[82] R. Bertran, A. Buyuktosunoglu, P. Bose, T. J. Slegel, G. Salem, S. Carey, R. F. Rizzolo, and T. Strach, “Voltage Noise in Multi-Core Processors: Empirical Characterization and Optimization Opportunities,” in *International Symposium on Microarchitecture (MICRO)*, Dec 2014, pp. 368–380.

[83] P. Pant and J. Zelman, “Understanding Power Supply Droop during At-Speed Scan Testing,” in *IEEE VLSI Test Symposium (VTS)*, May 2009, pp. 227–232.

[84] M. Tehranipoor and K. M. Butler, “Power Supply Noise: A Survey on Effects and Research,” *Des. Test. Comput.*, vol. 27, no. 2, pp. 51–67, March 2010.

[85] X. Lin, “Power Supply Droop and Its Impacts on Structural At-Speed Testing,” in *IEEE Asian Test Symposium (ATS)*, Nov 2012, pp. 239–244.

[86] R. Petersen, P. Pant, P. Lopez, A. Barton, J. Ignowski, and D. Josephson, “Voltage transient detection and induction for debug and test,” in *ITC*, Nov 2009, pp. 1–10.

[87] P. Mosalikanti, N. Kurd, C. Mozak, and T. Oshita, “Low Power Analog Circuit Techniques in the 5th Generation Intel Core Microprocessor (Broadwell),” in *IEEE Custom Integrated Circuits Conference (CICC)*, Sept 2015, pp. 1–4.

[88] H. Yu, Q. Xu, and P. H. W. Leong, “Fine-grained characterization of process variation in FPGAs,” in *International Conference on Field-Programmable Technology (FPT)*, Dec 2010, pp. 138–145.

[89] J. Wu, Z. Shi, and I. Y. Wang, “Firmware-only implementation of time-to-digital converter (TDC) in field-programmable gate array (FPGA),” in *Nuclear Science Symposium*, vol. 1, Oct 2003, pp. 177–181 Vol.1.

[90] Xilinx, “ML605 Hardware User Guide - UG534 (v1.8),” October 2012.

[91] “UCD9240 – Digital PWM System Controller,” Texas Instruments, 2008.

[92] D. Boneh, R. A. DeMillo, and R. J. Lipton, “On the Importance of Checking Cryptographic Protocols for Faults,” in *Advances in Cryptology — EUROCRYPT ’97*. Springer Berlin Heidelberg, 1997, pp. 37–51.

[93] R. Anderson and M. Kuhn, “Tamper resistance-a cautionary note,” in *Proceedings of the second Usenix workshop on electronic commerce*, vol. 2, 1997, pp. 1–11.

[94] E. Biham and A. Shamir, *Differential fault analysis of secret key cryptosystems.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 513–525. [Online]. Available: https://doi.org/10.1007/BFb0052259

[95] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *International Symposium on Computer Architecture (ISCA)*, ser. ISCA '14. Piscataway, NJ, USA: ACM/IEEE, 2014, pp. 361–372. [Online]. Available: http://dl.acm.org/citation.cfm?id=2665671.2665726

[96] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript," *CoRR*, vol. abs/1507.06955, 2015. [Online]. Available: http://arxiv.org/abs/1507.06955

[97] S. Endo, T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "An on-chip glitchy-clock generator for testing fault injection attacks," *J. Cryptogr. Eng.*, vol. 1, no. 4, pp. 265–270, 2011. [Online]. Available: http://dx.doi.org/10.1007/s13389-011-0022-y

[98] T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. E. Levin, T. D. Nguyen, and C. E. Irvine, "Moats and Drawbridges: An Isolation Primitive for Reconfigurable Hardware Based Systems," in *Symposium on Security and Privacy (S&P).* IEEE Computer Society, May 2007, pp. 281–295.

[99] T. Fuhr, E. Jaulmes, V. Lomne, and A. Thillard, "Fault Attacks on AES with Faulty Ciphertexts Only," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography.* IEEE, aug 2013.

[100] M. Mac and C. Wysocki, "Guaranteeing Silicon Performance with FPGA Timing Models," Intel Corporation, Tech. Rep., 2017. [Online]. Available: https://www.altera.com/en_US/pdfs/literature/wp/wp-01139-timing-model.pdf

[101] C. Beckhoff, D. Koch, and J. Torresen, "Short-Circuits on FPGAs Caused by Partial Runtime Reconfiguration," in *2010 International Conference on Field Programmable Logic and Applications.* IEEE, aug 2010.

[102] A. Moradi, M. T. M. Shalmani, and M. Salmasizadeh, "A Generalized Method of Differential Fault Attack Against AES Cryptosystem," in *Cryptographic Hardware and Embedded Systems - CHES 2006*, L. Goubin and M. Matsui, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 91–100.

[103] A. Amouri, J. Hepp, and M. Tahoori, "Self-heating thermal-aware testing of FPGAs," in *VLSI Test Symposium (VTS).* IEEE, Apr. 2014.

[104] M. Hutter and J.-M. Schmidt, "The Temperature Side Channel and Heating Fault Attacks," in *Smart Card Research and Advanced Applications.* Springer International Publishing, 2014, pp. 219–235.

[105] M. Zhao and G. E. Suh, “FPGA-Based Remote Power Side-Channel Attacks,” in *Symposium on Security and Privacy (S&P).* IEEE, May 2018, pp. 805–820. [Online]. Available: doi.ieeecomputersociety.org/10.1109/SP.2018.00049

[106] N. Selmane, S. Bhasin, S. Guilley, T. Graba, and J.-L. Danger, “WDDL is Protected against Setup Time Violation Attacks,” in *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC).* IEEE, sep 2009.

[107] S. Endo, Y. Li, N. Homma, K. Sakiyama, K. Ohta, and T. Aoki, “An Efficient Countermeasure against Fault Sensitivity Analysis Using Configurable Delay Blocks,” in *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography.* IEEE, sep 2012.

[108] A. Amouri and M. Tahoori, “A Low-Cost Sensor for Aging and Late Transitions Detection in Modern FPGAs,” in *International Conference on Field Programmable Logic and Applications (FPL).* IEEE, sep 2011, pp. 329–335.

[109] T. Sugawara, K. Sakiyama, S. Nashimoto, D. Suzuki, and T. Nagatsuka, “Oscillator without a Combinatorial Loop and its Threat to FPGA in Data Center,” *Electronics Letters*, vol. 55, no. 11, pp. 640–642, 2019.

[110] F. Schellenberg, “Novel methods of passive and active side-channel attacks,” doctoralthesis, Ruhr-Universität Bochum, Universitätsbibliothek, 2018.

[111] D. Bleichenbacher, “Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1,” in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 1462. Springer, 1998, pp. 1–12.

[112] P. C. Kocher, J. Jaffe, and B. Jun, “Differential Power Analysis,” in *Advances in cryptology—CRYPTO’99*, ser. LNCS, vol. 1666, Springer. Springer, 1999, pp. 789–789.

[113] L. Bossuet, M. Grand, L. Gaspar, V. Fischer, and G. Gogniat, “Architectures of Flexible Symmetric Key Crypto Engines—a Survey: From Hardware Coprocessor to Multi-crypto-processor System on Chip,” *ACM Comput. Surv.*, vol. 45, no. 4, pp. 41:1–41:32, Aug. 2013. [Online]. Available: http://doi.acm.org/10.1145/2501654.2501655

[114] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, “Designing and Implementing Malicious Hardware,” in *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[115] J. Rajendran, V. Jyothi, and R. Karri, “Blue team red team approach to hardware trust assessment,” in *ICCD.* IEEE Computer Society, 2011, pp. 285–288.

[116] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, “Stealthy Dopant-Level Hardware Trojans,” in *CHES*, ser. LNCS, vol. 8086. Springer, 2013, pp. 197–214.

[117] L. Lin, W. Burleson, and C. Paar, “MOLES: Malicious off-chip leakage enabled by side-channels,” in *ICCAD*. ACM, 2009, pp. 117–122.

[118] L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson, *Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering*. Berlin, Heidelberg: Springer, 2009, pp. 382–395. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04138-9_27

[119] M. Kasper, A. Moradi, G. T. Becker, O. Mischke, T. Güneysu, C. Paar, and W. Burleson, “Side channels as building blocks,” *J. Cryptographic Engineering*, vol. 2, no. 3, pp. 143–159, 2012. [Online]. Available: http://dx.doi.org/10.1007/s13389-012-0040-4

[120] S. Ghandali, G. T. Becker, D. Holcomb, and C. Paar, “A Design Methodology for Stealthy Parametric Trojans and Its Application to Bug Attacks,” in *CHES*, ser. LNCS, vol. 9813. Springer, 2016, pp. 625–647.

[121] R. Kumar, P. Jovanovic, W. P. Burleson, and I. Polian, “Parametric Trojans for Fault-Injection Attacks on Cryptographic Hardware,” in *FDTC*. IEEE Computer Society, 2014, pp. 18–28.

[122] Y. Shiyanovskii, F. G. Wolff, A. Rajendran, C. A. Papachristou, D. J. Weyer, and W. Clay, “Process reliability based trojans through NBTI and HCI effects,” in *AHS*. IEEE, 2010, pp. 215–222.

[123] S. Trimberger and S. McNeil, “Security of FPGAs in Data Centers,” in *IVSW*. IEEE Computer Society, 2017.

[124] J. D. Corbett, “The Xilinx Isolation Design Flow for Fault-Tolerant Systems,” 2012.

[125] I. Giechaskiel and K. Eguro, “Information Leakage Between FPGA Long Wires,” *CoRR*, vol. abs/1611.08882, 2016. [Online]. Available: http://arxiv.org/abs/1611.08882

[126] R. J. Masti, D. Rai, A. Ranganathan, C. Müller, L. Thiele, and S. Capkun, “Thermal Covert Channels on Multi-core Platforms,” in *USENIX Security Symposium*. USENIX Association, 2015, pp. 865–880.

[127] L. Zussa, I. Exurville, J.-M. Dutertre, J.-B. Rigaud, B. Robisson, A. Tria, and J. Clédière, “Evidence of an Information Leakage Between Logically Independent Blocks,” in *Proceedings of the Second Workshop on Cryptography and Security in Computing Systems*, ser. CS2 ’15. New York, NY, USA: ACM, 2015, pp. 25:25–25:30. [Online]. Available: http://doi.acm.org/10.1145/2694805.2694810

[128] T. D. Cnudde, B. Bilgin, B. Gierlichs, V. Nikov, S. Nikova, and V. Rijmen, “Does Coupling Affect the Security of Masked Implementations?” in *COSADE*, ser. LNCS, vol. 10348. Springer, 2017, pp. 1–18.

[129] C. Nagl, “Exploiting the Virtex 6 System Monitor for Power-Analysis Attacks,” Master’s thesis, IAIK - Graz University of Technology, 2012.

[130] C. O’Flynn and Z. Chen, “Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection,” *Journal of Cryptographic Engineering*, vol. 5, no. 1, pp. 53–69, Apr 2015. [Online]. Available: https://doi.org/10.1007/s13389-014-0087-5

[131] R. Turner, “System-level verification – a comparison of approaches,” in *International Workshop on Rapid System Prototyping*, Jul 1999, pp. 154–159.

[132] K. Chatterjee and D. Das, “Semiconductor Manufacturers’ Efforts to Improve Trust in the Electronic Part Supply Chain,” *IEEE Transactions on Components and Packaging Technologies*, vol. 30, no. 3, pp. 547–549, Sept 2007.

[133] S. P. Olarig and M. F. Angelo, “Method for the secure remote flashing of a BIOS memory,” Dec. 28 1999, uS Patent 6,009,524.

[134] A. Moradi, M. Kasper, and C. Paar, “On the Portability of Side-Channel Attacks - An Analysis of the Xilinx Virtex 4, Virtex 5, and Spartan 6 Bitstream Encryption Mechanism,” Cryptology ePrint Archive, Report 2011/391, 2011, https://eprint.iacr.org/2011/391.

[135] J. Hendricks and L. van Doorn, “Secure Bootstrap is Not Enough: Shoring Up the Trusted Computing Base,” in *SIGOPS European Workshop.* ACM, 2004. [Online]. Available: http://doi.acm.org/10.1145/1133572.1133600

[136] P. Maene, J. Götzfried, R. de Clercq, T. Müller, F. Freiling, and I. Verbauwhede, “Hardware-Based Trusted Computing Architectures for Isolation and Attestation,” *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 361–374, March 2018.

[137] A. Tang, S. Sethumadhavan, and S. Stolfo, “CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management,” in *USENIX Security Symposium.* Vancouver, BC: USENIX Association, 2017, pp. 1057–1074. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang

[138] M. G. Kuhn and R. J. Anderson, “Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations,” in *Information Hiding*, ser. Lecture Notes in Computer Science, vol. 1525. Springer, 1998, pp. 124–142.

[139] J. C. Wray, “An Analysis of Covert Timing Channels,” in *IEEE Symposium on Security and Privacy*, 1991, pp. 2–7.

[140] “Side-channel AttacK User Reference Architecture,” http://satoh.cs.uec.ac.jp/SAKURA/index.html.

[141] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1996.

[142] P. N. Whatmough, S. Das, Z. Hadjilambrou, and D. M. Bull, “Power Integrity Analysis of a 28 nm Dual-Core ARM Cortex-A57 Cluster Using an All-Digital Power Delivery Monitor,” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 6, pp. 1643–1654, June 2017.

[143] C. M. Medaglia and A. Serbanati, “An Overview of Privacy and Security Issues in the Internet of Things,” in *The Internet of Things*, D. Giusto, A. Iera, G. Morabito, and L. Atzori, Eds. New York, NY: Springer, 2010, pp. 389–395.

[144] R. Roman, P. Najera, and J. Lopez, “Securing the Internet of Things,” *Computer*, vol. 44, no. 9, pp. 51–58, Sep. 2011.

[145] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, “Understanding the Mirai Botnet,” in *USENIX Security Symposium*, 2017, pp. 1092–1110.

[146] C. Paar, A. Poschmann, S. Kumar, T. Eisenbarth, and L. Uhsadel, “A Survey of Lightweight-Cryptography Implementations,” *IEEE Design & Test of Computers*, vol. 24, pp. 522–533, Nov. 2007. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MDT.2007.178

[147] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, “Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers,” in *Proceedings of the Conference on Computer and Communications Security (CCS)*. ACM, October 2018.

[148] K. Gandolfi, C. Mourtel, and F. Olivier, “Electromagnetic analysis: Concrete results,” in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Ç. K. Koç, D. Naccache, and C. Paar, Eds., Springer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 251–261.

[149] J. A. Stankovic, “Wireless Sensor Networks,” *Computer*, vol. 41, no. 10, pp. 92–95, Oct. 2008.

[150] W. Shi and S. Dustdar, “The Promise of Edge Computing,” *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[151] Amazon-FreeRTOS, “Cloud-native IoT operating system for microcontrollers,” 2017-2018, https://github.com/aws/amazon-freertos. [Online]. Available: https://github.com/aws/amazon-freertos

[152] Microsoft, “Azure Device Catalog,” 2018, https://catalog.azureiotsolutions.com/. [Online]. Available: https://catalog.azureiotsolutions.com/

[153] C. Ramesh, S. B. Patil, S. N. Dhanuskodi, G. Provelengios, S. Pillement, D. Holcomb, and R. Tessier, “FPGA side channel attacks without physical access,” in *International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, May 2018, pp. paper–116.

[154] ARM MBED, “SSL Library mbed TLS / PolarSSL,” 2008-2016, https://tls.mbed.org/.

[155] R. Barry and A. W. S. (AWS), “FreeRTOS,” 2003–2019, https://www.freertos.org/.

[156] J. Jaffe, “A First-Order DPA Attack Against AES in Counter Mode with Unknown Initial Counter,” in *Cryptographic Hardware and Embedded Systems (CHES)*, P. Paillier and I. Verbauwhede, Eds., Berlin, Heidelberg, 2007, pp. 1–13.

[157] J. Longo, E. De Mulder, D. Page, and M. Tunstall, “SoC It to EM: ElectroMagnetic Side-Channel Attacks on a Complex System-on-Chip,” in *Cryptographic Hardware and Embedded Systems – CHES 2015*, T. Güneysu and H. Handschuh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 620–640.

[158] C. D. Walter, “Sliding windows succumbs to Big Mac attack,” in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2001, pp. 286–299.

[159] W. Schindler and K. Itoh, “Exponent Blinding Does Not Always Lift (Partial) Spa Resistance to Higher-Level Security,” in *Applied Cryptography and Network Security*, J. Lopez and G. Tsudik, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 73–90.

[160] G. Perin and Ł. Chmielewski, “A Semi-Parametric Approach for Side-Channel Attacks on Protected RSA Implementations,” in *Smart Card Research and Advanced Applications*, N. Homma and M. Medwed, Eds. Cham: Springer International Publishing, 2016, pp. 34–53.

[161] D. Genkin, A. Shamir, and E. Tromer, *RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis*. Berlin, Heidelberg: Springer, 2014, pp. 444–461. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44371-2_25

[162] T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. M. Shalmani, “On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme,” in *Advances in Cryptology – CRYPTO 2008*, D. Wagner, Ed. Berlin, Heidelberg: Springer, 2008, pp. 203–220.

[163] J. Balasch, B. Gierlichs, R. Verdult, L. Batina, and I. Verbauwhede, “Power analysis of Atmel CryptoMemory – recovering keys from secure EEPROMs,” in *Cryptographers’ Track at the RSA Conference*. Springer, 2012, pp. 19–34.

[164] A. A. Cardenas, T. Roosta, and S. Sastry, “Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems,” *Ad Hoc Networks*, vol. 7, no. 8, pp. 1434–1447, 2009.

[165] R. Langner, “Stuxnet: Dissecting a cyberwarfare weapon,” *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[166] C. Percival, "Cache missing for fun and profit," 2005.

[167] Y. Yarom and K. Falkner, "FLUSH+ RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack." in *USENIX Security Symposium*, 2014, pp. 719–732.

[168] M. Seaborn and T. Dullien, "Exploiting the DRAM rowhammer bug to gain kernel privileges," *Black Hat*, vol. 15, 2015.

[169] V. Van Der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic rowhammer attacks on mobile platforms," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security.* ACM, 2016, pp. 1675–1689.

[170] S. Bhattacharya and D. Mukhopadhyay, "Curious case of rowhammer: flipping secret exponent bits using timing analysis," in *International Conference on Cryptographic Hardware and Embedded Systems.* Springer, 2016, pp. 602–624.

[171] Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One bit flips, one cloud flops: Cross-vm row hammer attacks and privilege escalation," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 19–35.

[172] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip Feng Shui: Hammering a Needle in the Software Stack," in *USENIX Security Symposium.* Austin, TX: USENIX Association, August 2016.

[173] D. Gruss, M. Lipp, M. Schwarz, D. Genkin, J. Juffinger, S. O'Connell, W. Schoechl, and Y. Yarom, "Another flip in the wall of rowhammer defenses," in *2018 IEEE Symposium on Security and Privacy (SP).* IEEE, 2018, pp. 245–261.

[174] M. Lipp, M. T. Aga, M. Schwarz, D. Gruss, C. Maurice, L. Raab, and L. Lamster, "Nethammer: Inducing rowhammer faults through network requests," *arXiv preprint arXiv:1805.04956*, 2018.

[175] A. Tatar, R. K. Konoth, E. Athanasopoulos, C. Giuffrida, H. Bos, and K. Razavi, "Throwhammer: Rowhammer attacks over the network and defenses," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, 2018, pp. 213–226.

[176] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based Fault Injection Attacks against Intel SGX," in *IEEE Symposium on Security and Privacy (S&P'20)*, 2020.

[177] Z. Kenjar, T. Frassetto, D. Gens, M. Franz, and A.-R. Sadeghi, "V0LTpwn: Attacking x86 Processor Integrity from Software," *arXiv preprint arXiv:1912.04870*, 2019.

[178] P. Qiu, D. Wang, Y. Lyu, and G. Qu, "VoltJockey: Breaching TrustZone by Software-Controlled Voltage Manipulation over Multi-core Frequencies," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2019, pp. 195–209.

[179] D. Mahmoud and M. Stojilović, “Timing violation induced faults in multi-tenant FPGAs,” in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE).* IEEE, 2019, pp. 1745–1750.

[180] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, “RAM-Jam: Remote Temperature and Voltage Fault Attack on FPGAs using Memory Collisions,” in *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, Aug 2019, pp. 48–55.

[181] G. Provelengios, D. Holcomb, and R. Tessier, “Characterizing power distribution attacks in multi-user FPGA environments,” in *International Conference on Field Programmable Logic and Applications (FPL).* IEEE, 2019, pp. 194–201.

[182] W. Hunter, C. McCarty, and L. Lerner, “Improved Techniques for Sensing Intra-Device Side Channel Leakage,” in *International Symposium on Field-Programmable Custom Computing Machines (FCCM).* IEEE, 2019, pp. 328–328.

[183] J. Gravellier, J. Dutertre, Y. Teglia, and P. Loubet-Moundi, “High-Speed Ring Oscillator based Sensors for Remote Side-Channel Attacks on FPGAs,” in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec 2019, pp. 1–8.

[184] J. Gravellier, J.-M. DUTERTRE, Y. Teglia, P. Loubet-Moundi, and O. Francis, “Remote Side-Channel Attacks on Heterogeneous SoC,” in *Smart Card Research and Advanced Applications, 18th International Conference, CARDIS 2019*, Pragues, Czech Republic, Nov. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02380092

[185] J.-M. Schmidt, T. Plos, M. Kirschbaum, M. Hutter, M. Medwed, and C. Herbst, “Side-channel leakage across borders,” in *International Conference on Smart Card Research and Advanced Applications.* Springer, 2010, pp. 36–48.

[186] C. O’Flynn and A. Dewar, “On-Device Power Analysis Across Hardware Security Domains,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 126–153, 2019.

[187] J. Krautter, D. R. E. Gnad, F. Schellenberg, A. Moradi, and M. B. Tahoori, “Active Fences against Voltage-based Side Channels in Multi-Tenant FPGAs,” in *Proceedings of the International Conference on Computer-Aided Design (ICCAD).* ACM, Nov. 2019.

[188] D. R. E. Gnad, S. Rapp, J. Krautter, and M. B. Tahoori, “Checking for Electrical Level Security Threats in Bitstreams for Multi-Tenant FPGAs,” in *International Conference on Field-Programmable Technology (FPT).* Naha, Japan: IEEE, Dec. 2018.

[189] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, “Mitigating Electrical-Level Attacks towards Secure Multi-Tenant FPGAs in the Cloud,” *ACM Trans.*

Reconfigurable Technol. Syst., vol. 12, no. 3, Aug. 2019. [Online]. Available: https://doi.org/10.1145/3328222

[190] Altera Corporation, "AN 567: Quartus II Design Separation Flow," 2009.

[191] I. Giechaskiel, K. Eguro, and K. B. Rasmussen, "Leakier Wires: Exploiting FPGA Long Wires for Covert- and Side-Channel Attacks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 3, Aug. 2019. [Online]. Available: https://doi.org/10.1145/3322483

[192] G. Provelengios, C. Ramesh, S. B. Patil, K. Eguro, R. Tessier, and D. Holcomb, "Characterization of long wire data leakage in deep submicron FPGAs," in *International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2019, pp. 292–297.

Acronyms

ADC Analog-to-Digital Converter. 10, 119, 123
AES Advanced Encryption Standard. 15, 16, 53–55, 57, 60, 61, 63, 65–72, 81, 82, 84, 85, 147, 149, 150
ALM Adaptive Logic Module; an advanced configurable LUT, used in most Intel FPGAs. 64, 65
BRAM Block RAM; typically SRAM memory, available in user-configurable FPGA logic. 80, 82, 118
CPA Correlation Power Analysis. 13, 14, 82, 84, 85, 92, 100, 119
DCM Digital Clock Manager. 81
DFA Differential Fault Analysis. 6, 15, 53–55, 57, 60, 63, 66–71, 149, 150
DoS Denial-of-Service Attack; compromises *Availability* of an asset. 5, 6, 12, 53, 72, 119
DPA Differential Power Analysis. 75
FPGA Field Programmable Gate Array. 3, 4, 7, 8, 11, 12, 15, 53–60, 62–66, 69–72, 118, 123, 149
GPU Graphics Processing Unit. 3
HDL Hardware Description Language; class of computer languages used to design electronic circuits, major contenders are VHDL and Verilog. 57
IC Integrated Circuit. 75
ILA Integrated Logic Analyzer; available Debug Core in Xilinx FPGAs. 80, 150
LAB Logic Array Block; block containing various configurable elements and local interconnect in Intel FPGAs. 65
LSB Least Significant Bit. 59
LUT Look-Up Table; used in most FPGAs to implement custom logic. 57, 58, 64–67, 69, 72, 145, 149
MSB Most Significant Bit. 59
PCB Printed Circuit Board. 5, 7, 86, 88, 118
PDN Power Distribution Network. 7, 56, 57, 71, 75–78, 85, 86, 119, 120, 150
RO Ring Oscillator; combinational logic with a feedback loop that leads to oscillation. 11, 56–67, 70–73, 118, 149
RSA a common public-key cryptosystem. 75
SCA Side-Channel Analysis. 75–77, 86, 87, 150
SoC System-on-Chip. 54, 71, 76, 77, 150
SPA Simple Power Analysis. 92
SPN Substitution-Permutation Network. 15
STA Static Timing Analysis. 64, 67, 69, 70, 155
TCB Trusted Computing Base. 88
TDC Time-to-Digital Converter. 11, 12, 78, 79, 92, 118, 147, 150

List of Figures

List of Tables

A. Appendix on FPGAhammer

A.1. Single Ring Oscillator Verilog Source Code

```
module osc ( enablein, dummyout );
  input enablein;
  output dummyout;
  wire enablein_lut;
  wire loop_lut, loop;
  lut_input enable_lutin(enablein, enablein_lut);
  lut_input loop_lutin(loop, loop_lut);
  lut_output loop_lutout(~loop_lut & enablein_lut, loop);
  assign dummyout = loop;
endmodule
```

Listing A.1: Single RO module Verilog source code using low-level primitives to implement a two-input NAND gate

A.2. Ring Oscillator Grid Generation Verilog Source Code

```
module osc_array ( clkin, enablein, rstin, dummyout, [...] );
  parameter amount = 10000;
  [...]
  reg enable_oscs = 1'b0;
  wire enable_oscs_g;
  //Global clock buffer routing:
  global enable_oscs_glob (.in(enable_oscs), .out(enable_oscs_g));
  output [amount-1:0] dummyout;
  genvar i;
  generate
    for (i=0; i < amount; i=i+1) begin : oscs_gen
      osc osc_inst(.enablein(enable_oscs_g), .dummyout(dummyout[i]));
    end
  endgenerate
  [...]
endmodule
```

Listing A.2: Relevant parts of the RO grid generation Verilog source code with global clock buffer routing and dummy output signals

A.3. Top-Level RO Instantiation Verilog Source Code

```
parameter ro_amount = 10000;
output [ro_amount-1:0] dummyout /* synthesis noprune=1
        altera_attribute="-name VIRTUAL_PIN ON" */;

osc_array osc_array_inst ( .clkin(clk), .rstin(rst_n),
        .enablein(enable_oscs), .dummyout(dummyout), [...] );

defparam osc_array_inst.amount = ro_amount;
```

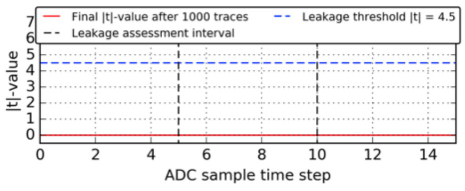
Listing A.3: Instantiation of ROs with virtual pins in the top-level module; To generate bare ROs without virtual pins, the *dummyout* output declaration is removed and the respective port of the *osc_array* module instance left unconnected

B. Appendix on Leaky Noise

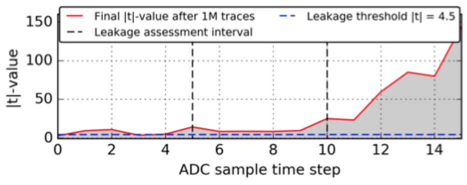
B.1. Results of all Leakage Assessments

B.1.1. ESP32-devkitC

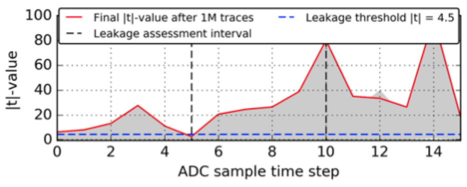
B.1.1.1. AES



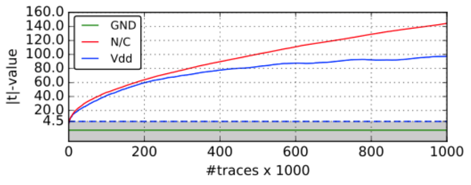
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1k traces collected during AES encryptions on the ESP32-devkitC with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the ESP32-devkitC with the ADC pin disconnected (N/C).



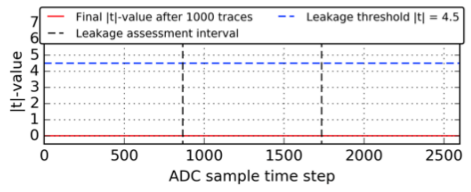
(c) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the ESP32-devkitC with the ADC pin connected to V_{dd}.



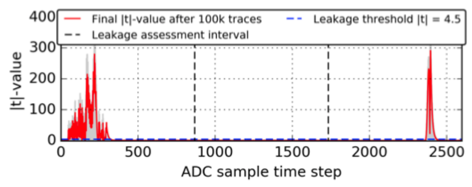
(d) Leakage Assessment progress on mbedTLS AES with {Vdd, GND, N/C} connected to the ADC on the ESP32-devkitC.

Figure B.1.: Results of Leakage Assessments on ESP32-devkitC for mbedTLS AES.

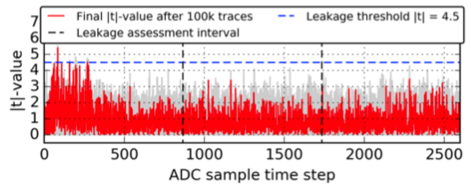
B.1.1.2. Sliding Window Modular Exponentiation



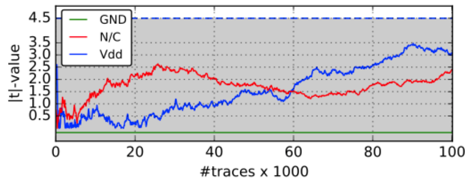
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1k traces collected during modular exponentiation on the ESP32-devkitC with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the ESP32-devkitC with the ADC pin disconnected (N/C).



(c) First order leakage assessment result based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the ESP32-devkitC with the ADC connected to V_{dd}.

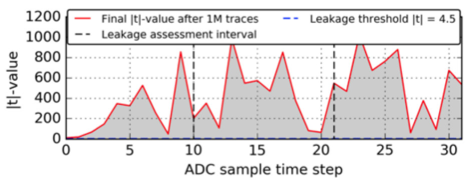


(d) Leakage Assessment progress on mbedTLS modular exponentiation with {Vdd, GND, N/C} connected to the ADC on the ESP32-devkitC.

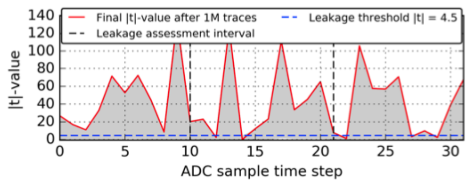
Figure B.2.: Results of Leakage Assessments on ESP32-devkitC for mbedTLS modular exponentiation.

B.1.2. STM32F407VG Discovery #1

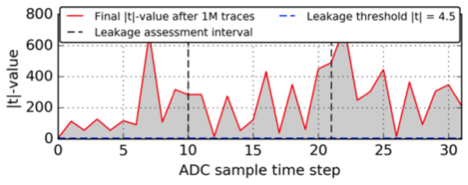
B.1.2.1. AES



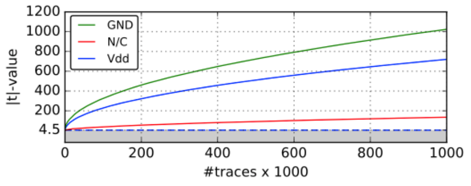
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #1 with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #1 with the ADC pin disconnected (N/C).



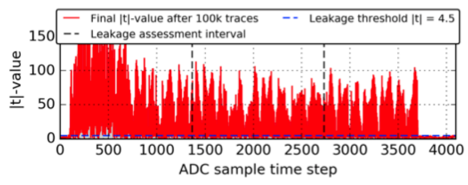
(c) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #1 with the ADC pin connected to V_{dd}.



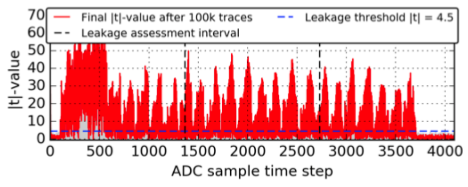
(d) Leakage Assessment progress on mbedTLS AES with {Vdd, GND, N/C} connected to the ADC on the STM32F407VG Discovery #1.

Figure B.3.: Results of Leakage Assessment on STM32F407VG Discovery #1 for mbedTLS AES.

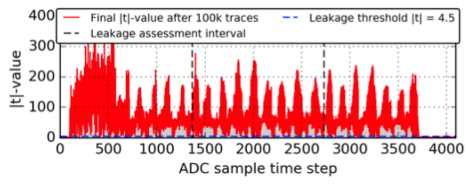
B.1.2.2. Sliding Window Modular Exponentiation



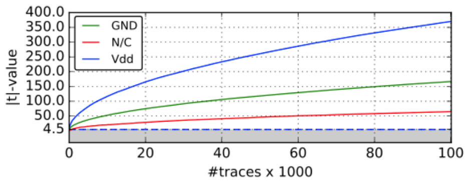
(a) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #1 with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #1 with the ADC pin disconnected (N/C).



(c) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #1 with the ADC pin connected to V_{dd}.

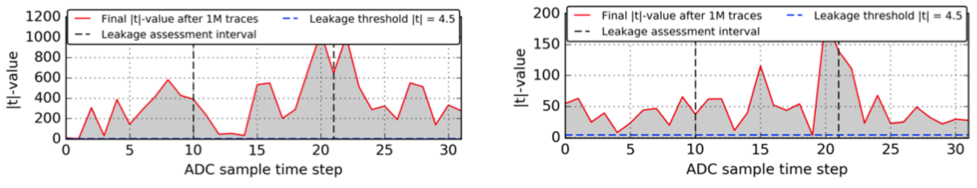


(d) Leakage Assessment progress on mbedTLS modular exponentiation with {Vdd, GND, N/C} connected to the ADC on the STM32F407VG Discovery #1.

Figure B.4.: Results of Leakage Assessments on STM32F407VG Discovery #1 for mbedTLS modular exponentiation.

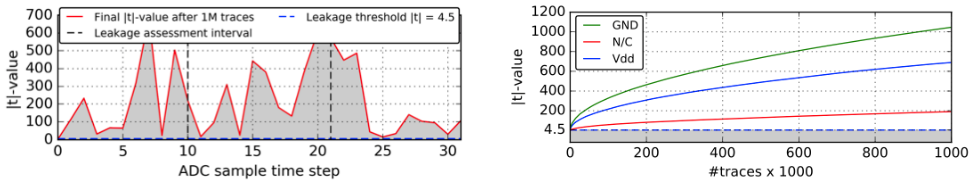
B.1.3. STM32F407VG Discovery #2

B.1.3.1. AES



(a) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #2 with the ADC pin connected to GND.

(b) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #2 with the ADC pin disconnected (N/C).

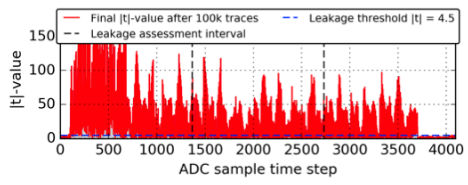


(c) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32F407VG Discovery #2 with the ADC pin connected to V_{dd}.

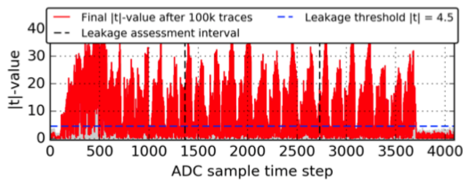
(d) Leakage Assessment progress on mbedTLS AES with {Vdd, GND, N/C} connected to the ADC on the STM32F407VG Discovery #2.

Figure B.5.: Results of Leakage Assessments on STM32F407VG Discovery #2 for mbedTLS AES.

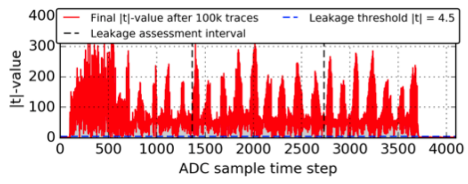
B.1.3.2. Sliding Window Modular Exponentiation



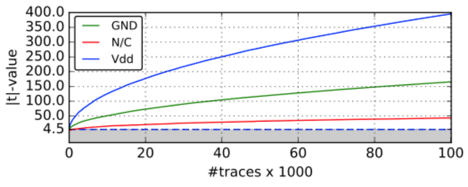
(a) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #2 with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #2 with the ADC pin disconnected (N/C).



(c) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32F407VG Discovery #2 with the ADC pin connected to V_{dd}.

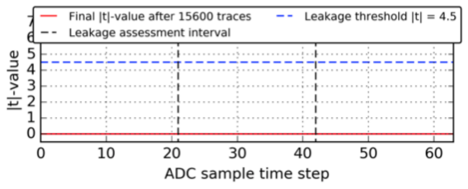


(d) Leakage Assessment progress on mbedTLS modular exponentiation with {Vdd, GND, N/C} connected to the ADC on the STM32F407VG Discovery #2.

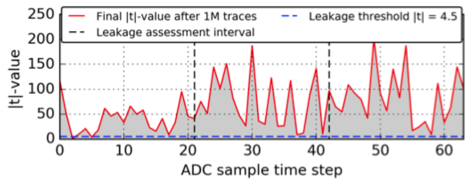
Figure B.6.: Results of Leakage Assessments on STM32F407VG Discovery #2 for mbedTLS modular exponentiation.

B.1.4. STM32L475 IoT Node

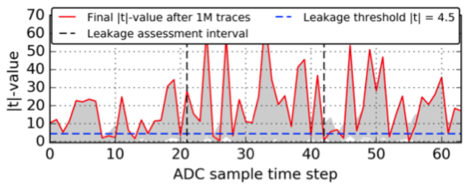
B.1.4.1. AES



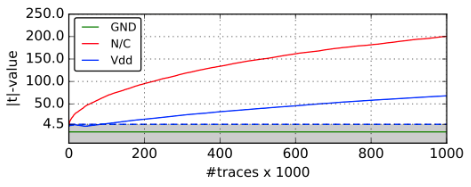
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32L475 IoT Node with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32L475 IoT Node with the ADC pin disconnected (N/C).



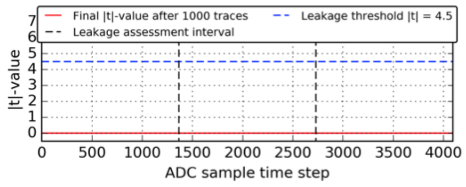
(c) First order leakage assessment results based on a fixed-vs-random t-test for 1M traces collected during AES encryptions on the STM32L475 IoT Node with the ADC pin connected to V_{dd}.



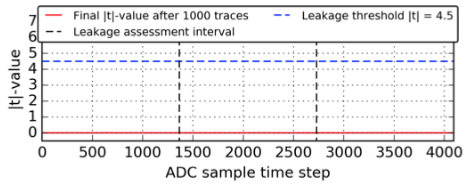
(d) Leakage Assessment progress on mbedTLS AES with {Vdd, GND, N/C} connected to the ADC on the STM32L475 IoT Node.

Figure B.7.: Results of Leakage Assessments on STM32L475 IoT Node for mbedTLS AES.

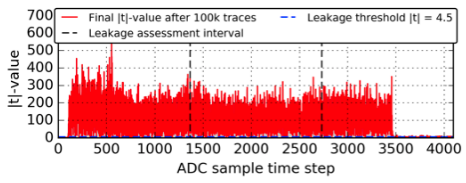
B.1.4.2. Sliding Window Modular Exponentiation



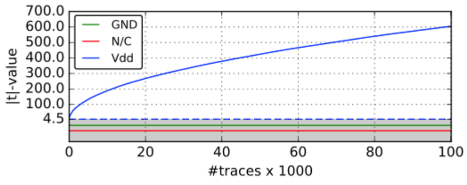
(a) First order leakage assessment results based on a fixed-vs-random t-test for 1k traces collected during modular exponentiation on the STM32L475 IoT Node with the ADC pin connected to GND.



(b) First order leakage assessment results based on a fixed-vs-random t-test for 1k traces collected during modular exponentiation on the STM32L475 IoT Node with the ADC pin disconnected (N/C).



(c) First order leakage assessment results based on a fixed-vs-random t-test for 100k traces collected during modular exponentiation on the STM32L475 IoT Node with the ADC pin connected to V_{dd}.

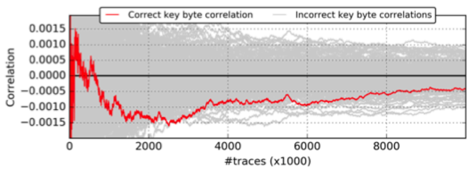


(d) Leakage Assessment progress on mbedTLS modular exponentiation with {Vdd, GND, N/C} connected to the ADC on the STM32L475 IoT Node.

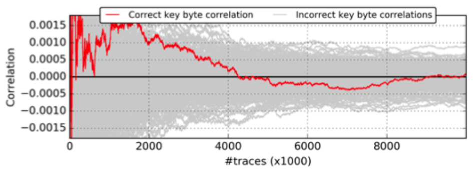
Figure B.8.: Results of Leakage Assessments on STM32L475 IoT Node for mbedTLS modular exponentiation.

B.2. Results of CPA for all secret AES key bytes on Vdd and GND

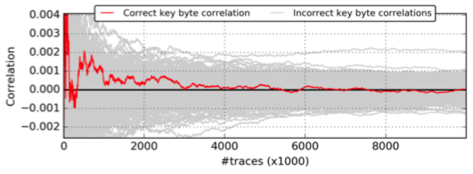
B.2.1. ADC pin connected to GND



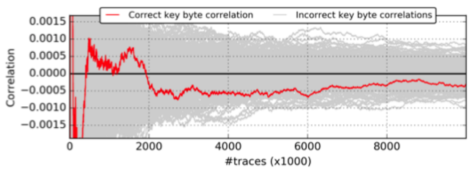
(a) CPA progress for the 0th secret key byte



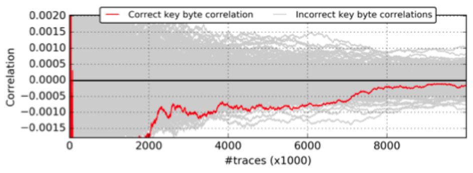
(b) CPA progress for the 1st secret key byte



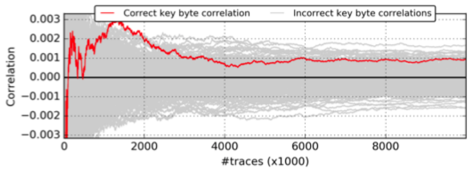
(c) CPA progress for the 2nd secret key byte



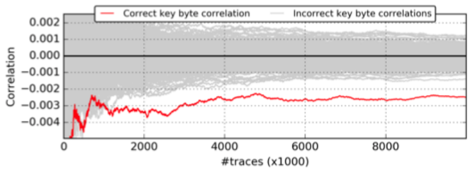
(d) CPA progress for the 3rd secret key byte



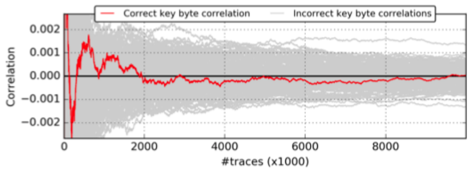
(e) CPA progress for the 4th secret key byte



(f) CPA progress for the 5th secret key byte

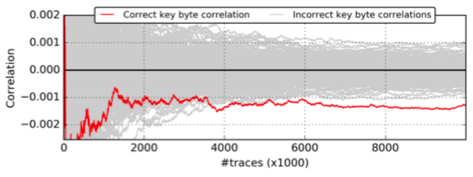


(g) CPA progress for the 6th secret key byte

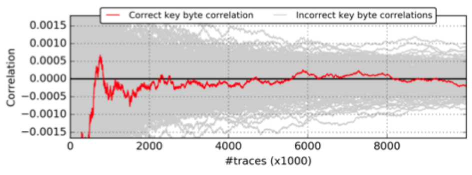


(h) CPA progress for the 7th secret key byte

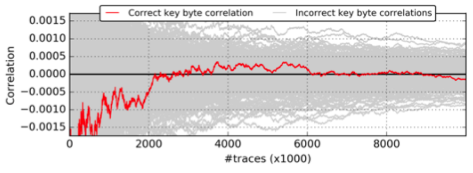
Figure B.9.: Results of a CPA attack on the last secret round key (bytes 0 to 7) of AES on the STM32F407VG Discovery #1 @168MHz with the ADC connected to GND and the program compiled with the -Os optimization option. Each plot shows the correlation progress of all 256 key candidates for a specific key byte over $10M$ traces and the respective correct key candidate is marked red.



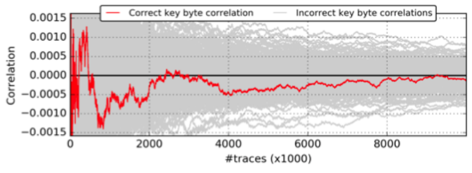
(a) CPA progress for the 8th secret key byte



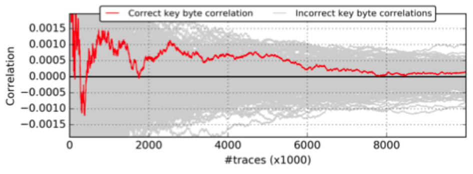
(b) CPA progress for the 9th secret key byte



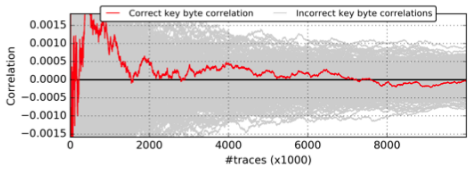
(c) CPA progress for the 10th secret key byte



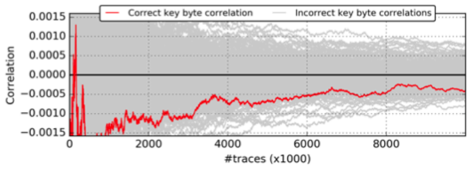
(d) CPA progress for the 11th secret key byte



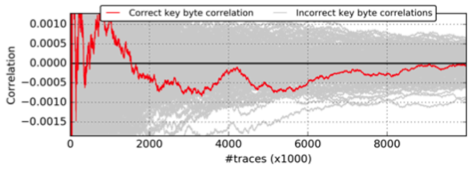
(e) CPA progress for the 12th secret key byte



(f) CPA progress for the 13th secret key byte



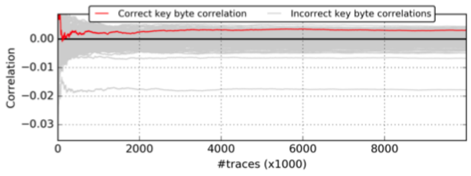
(g) CPA progress for the 14th secret key byte



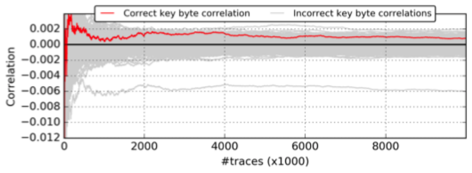
(h) CPA progress for the 15th secret key byte

Figure B.10.: Results of a CPA attack on the last secret round key (bytes 8 to 15) of AES on the STM32F407VG Discovery #1 @168MHz with the ADC connected to GND and the program compiled with the -Os optimization option. Each plot shows the correlation progress of all 256 key candidates for a specific key byte over $10M$ traces and the respective correct key candidate is marked red.

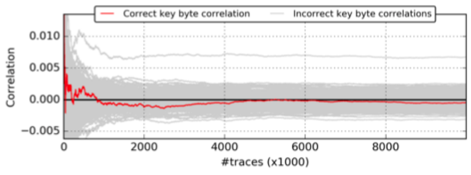
B.2.2. ADC pin connected to Vdd



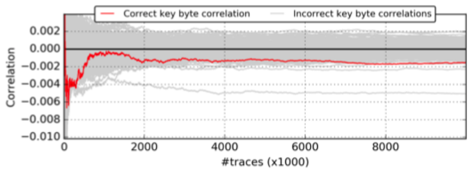
(a) CPA progress for the 0th secret key byte



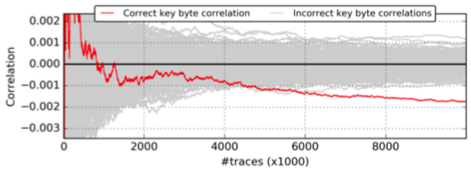
(b) CPA progress for the 1st secret key byte



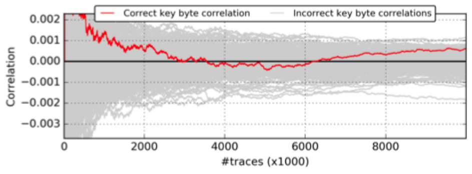
(c) CPA progress for the 2nd secret key byte



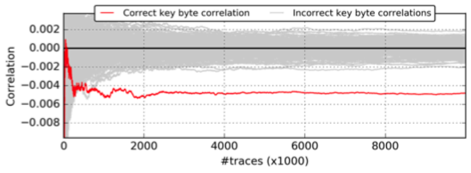
(d) CPA progress for the 3rd secret key byte



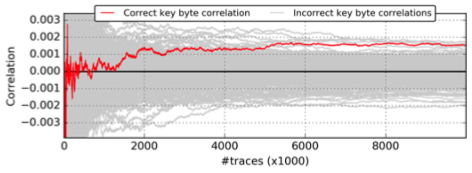
(e) CPA progress for the 4th secret key byte



(f) CPA progress for the 5th secret key byte

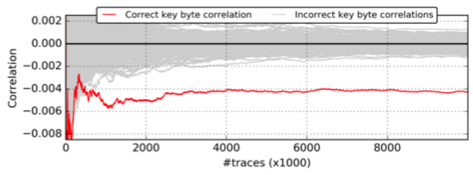


(g) CPA progress for the 6th secret key byte

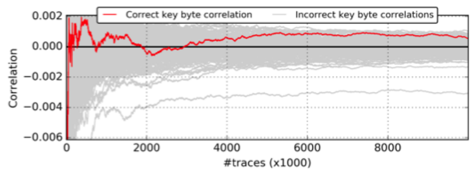


(h) CPA progress for the 7th secret key byte

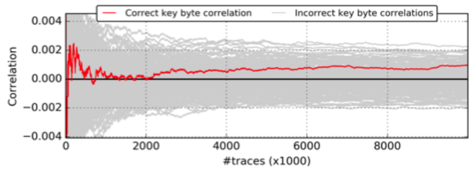
Figure B.11.: Results of a CPA attack on the last secret round key (bytes 0 to 7) of AES on the STM32F407VG Discovery #2 @56MHz with the ADC connected to Vdd and the program compiled with the -O0 optimization option. Each plot shows the correlation progress of all 256 key candidates for a specific key byte over $10M$ traces and the respective correct key candidate is marked red.



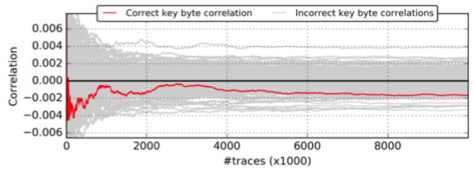
(a) CPA progress for the 8th secret key byte



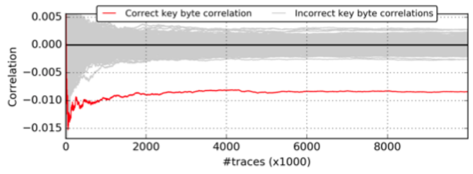
(b) CPA progress for the 9th secret key byte



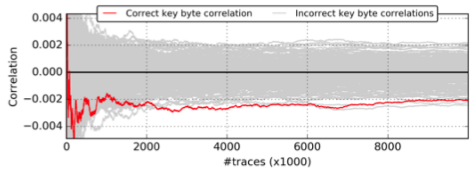
(c) CPA progress for the 10th secret key byte



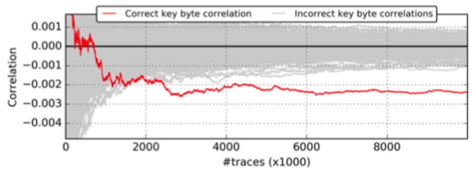
(d) CPA progress for the 11th secret key byte



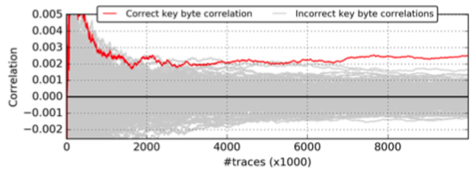
(e) CPA progress for the 12th secret key byte



(f) CPA progress for the 13th secret key byte



(g) CPA progress for the 14th secret key byte



(h) CPA progress for the 15th secret key byte

Figure B.12.: Results of a CPA attack on the last secret round key (bytes 8 to 15) of AES on the STM32F407VG Discovery #2 @168MHz with the ADC connected to Vdd and the program compiled with the -O0 optimization option. Each plot shows the correlation progress of all 256 key candidates for a specific key byte over $10M$ traces and the respective correct key candidate is marked red.

B.3. Simplified Source Code of the Experiments

Listing B.1: generic adcTask to sample the ADC for traces

```c
// adcHandle = ..
void adcTask(void * pvParameters) {
  while(1) {
    // wait for notify from mbedTask:
    while (ulTaskNotifyTake(pdTRUE, portMAX_DELAY) == 0);
    adc_get_samples(); // see details below (esp32: CPU, stm32: DMA)
    uart_send(adc_data, sizeof(adc_data));
    xTaskNotifyGive( mbedHandle ); // notify mbedTask
  }
}
```

Listing B.2: generic mbedTask to run mbedTLS AES or modular exponentiation

```c
// mbedHandle = ..
void mbedTask(void * pvParameters) {
  // [...] init contexts/secrets for mbedtls here
  while(1) {
    // read message from uart
    uart_read(msg, sizeof(msg));
#ifdef EXP
    mbedtls_mpi_read_string(&g, 16, msg)
#endif
    // start ADC in other task:
    xTaskNotifyGive( adcHandle ); // notify adcTask
#ifdef EXP
    mbedtls_mpi_exp_mod(&dummy, &g, &e, &modulus, NULL)
#else // AES
    mbedtls_internal_aes_encrypt(&ctx, msg, dummy);
#endif
    // wait for notify from adcTask:
    while (ulTaskNotifyTake(pdTRUE, portMAX_DELAY) == 0);
  }
}
```

Listing B.3: adc_get_samples for ESP32 CPU Task-based

```c
static inline void adc_get_samples() {
  for (int i=0;i<ADC_WORDS;i++) {
    adc_data[i] = adc1_get_raw(ADC_CHAN_SEL);
  }
}
```

Listing B.4: adc_get_samples for ESP32 ULP-based, plus ULP assembly code (adc.S). Please note, in the actual implementation we directly used the ULP from the mbedTask instead of a separate adcTask.

```c
static inline void adc_get_samples() {
  adc1_ulp_enable();
  ulp_load_binary(0, ulp_main_bin_start,ulp_bin_size);
  ulp_set_wakeup_period(0, 1000);
  while(((volatile typeof(ulp_sync_back)) ulp_sync_back) == 0);
  *((volatile typeof(ulp_sync_back)*) &ulp_sync_back) = 0;
  uint32_t* p_ulp_adc_data = &ulp_adc_data;
  for (int i=0;i<ADC_WORDS;i++) {
    adc_data[i] = (uint16_t) (*p_ulp_adc_data++) & 0xffff;
  }
}

adc.S:
    move r0, adc_data
  measure:
    adc r2, adc_nr, adc_channel + 1
    st r2,r0,0
    add r0, r0, 1
    jumpr measure, adc_data+ADC_WORDS, lt
    // sync back to main cpu, which spinlocks:
    move r1, sync_back
    move r2, 0x0001
    st r2, r1,0
    halt
```

Listing B.5: adc_get_samples for STM32 and ADC-DMA interrupt handler

```
static inline void adc_get_samples()
{
  // start to acquire ADC samples through DMA
  HAL_ADC_Start_DMA(&hadc3, adc_data, ADC_WORDS)
  // wait for ADC/DMA to finish while mbedTask executes
  osSignalWait(0x0001, osWaitForever);
  // make sure DMA is stopped
  HAL_ADC_Stop_DMA(&hadc3);
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
  osSignalSet(adcHandle, 0x0001);
}
```