Karlsruher Institut für Technologie

Design and Implementation of Physically Secure Reconfigurable System

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

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

genehmigte

Dissertation

von

Sergej Meschkov

geb. in Rostow-am-Don

Tag der mündlichen Prüfung: 13. Mai 2026

1. Referent/Referentin:	Prof. Dr. Mehdi B. Tahoori, Karlsruher Institut für Technologie (KIT)
2. Referent/Referentin:	Prof. Dr.-Ing. Elif Bilge Kavun, Technische Universität Dresden (TUD)

Acknowledgements

First and foremost, I would like to thank my advisor, Prof. Dr. Mehdi B. Tahoori, for his outstanding supervision, enthusiasm for research, and continual readiness to discuss ideas and provide constructive feedback. His guidance has been instrumental to the development and completion of this dissertation.

I am deeply grateful to all my colleagues at the Chair of Dependable Nano Computing for creating a stimulating, friendly, and collaborative working environment. The exchanges, joint problem-solving, and the many small and large contributions from everyone around me have enriched this work more than I can describe.

I would like to give special thanks to more senior colleagues who supported me in the early stages of my research and helped me with writing my first papers: Dr. Jonas Krautter and Dr. Christopher Münch. In particular, I am deeply grateful to Dr. Dennis Gnad, who not only assisted immensely during those formative research steps but also supervised my Master's thesis at the Chair. Their advice, careful reading, and practical help were invaluable in getting my research and publications off the ground.

I would also like to thank Prof. Dr. Amir Moradi (Technical University of Darmstadt) and Daniel Lammers (Ruhr University Bochum) for a long-running cooperation. Our sustained collaboration directly contributed to several key results and produced some of the most valuable research presented in this dissertation, not to mention the many insightful discussions we shared.

Finally, I owe my greatest thanks to my family for their love and patience throughout this journey. In particular, I want to thank my wife, Anna, for her constant encouragement and practical support, and my sons, Konstantin and Alexander, for bringing joy and perspective to my life. Without their support, I would not have been able to complete this dissertation.

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 habe 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.

Maulbronn, 15. März 2026
Sergej Meschkov

Abstract

The escalating incidence of hardware-level breaches highlights the critical necessity for robust physical security architectures. Among these threats, Side-Channel Analysis (SCA) and Fault Injection (FI) attacks present formidable challenges to security-critical applications, particularly those operating in adversarial environments. This thesis addresses fundamental vulnerabilities in the hardware root-of-trust by characterizing novel side-channel leakage vectors in established infrastructures and engineering automated, resource-efficient frameworks for the deployment of provably secure reconfigurable hardware.

The first part of the thesis challenges established physical security paradigms through the introduction and empirical validation of novel threat models targeting cryptographic devices in both test and operational contexts. It introduces a threat model where Design-for-Testability (DfT) infrastructure, specifically timing and delay test data from publicly accessible outputs, is utilized to mount template-based side-channel attacks. Furthermore, the research characterizes Jitter-based Side-Channel Analysis (JitSCA), a novel attack vector that exploits signal jitter in communication links. It is empirically demonstrated that jitter side-channel leakage propagates across galvanically isolated boundaries allowing full cryptographic key recovery.

The second part of the thesis focuses on the systematic automation of secure hardware design for reconfigurable platforms. To address the prohibitive overhead and complexity of manual countermeasure integration on Field Programmable Gate Arrays (FPGAs), the AGEMA_FPGA Electronic Design Automation (EDA) flow is proposed. This framework optimizes the mapping of specific first-order masked gadgets specifically for FPGA architectures by decomposing non-linear functions to align with intrinsic Look-Up Table (LUT) boundaries while maintaining provably secure masking properties.

Finally, the thesis presents a security-first, open-source reconfigurable fabric and toolchain. By utilizing secure, composable gadgets and an automated mapping flow, the framework enables the implementation of SCA- and FI-resilient circuits without requiring domain-specific expertise. This approach achieves area reduction over secure implementations deployed on a baseline reconfigurable architecture, with physical security validated through Test Vector Leakage Assessment (TVLA) over 100 million traces. Collectively, these contributions bridge the gap between theoretical hardware vulnerabilities and practical, scalable defenses, establishing a flexible and verifiable foundation for modern hardware roots-of-trust.

Zusammenfassung

Die zunehmende Häufigkeit von Angriffen auf Hardware-Ebene unterstreicht die Notwendigkeit robuster Paradigmen für die physische Sicherheit. Unter diesen Bedrohungen stellen Seitenkanalangriffe (SCA) und Fehlerangriffe (FI) gewaltige Herausforderungen für sicherheitskritische Anwendungen dar, insbesondere für solche, die in nicht vertrauenswürdigen Umgebungen operieren. Diese Dissertation befasst sich mit grundlegenden Schwachstellen in der Hardware-Root-of-Trust, indem sie neuartige Seitenkanäle in etablierten Infrastrukturen charakterisiert und automatisierte, ressourceneffiziente Frameworks für den Einsatz von nachweislich sicherer, rekonfigurierbarer Hardware entwickelt.

Der erste Teil der Dissertation stellt etablierte Paradigmen der physischen Sicherheit in Frage, indem er neue Bedrohungsmodelle einführt und empirisch validiert. Es wird ein formales Bedrohungsmodell eingeführt, bei dem die Testinfrastruktur (DfT), speziell Timing- und Verzögerungstestdaten von öffentlich zugänglichen Ausgängen, genutzt werden, um template-basierte Seitenkanalangriffe durchzuführen. Darüber hinaus wird JitSCA charakterisiert, ein neuartiger Angriffsvektor, der Signal-Jitter in Kommunikationsverbindungen ausnutzt. Es wird empirisch nachgewiesen, dass Jitter-Seitenkanal über galvanisch getrennte Grenzen hinweg propagiert und eine vollständige Wiederherstellung kryptografischer Schlüssel ermöglicht.

Der zweite Teil der Dissertation konzentriert sich auf die systematische Automatisierung des sicheren Hardware-Designs für rekonfigurierbare Plattformen. Um den unerschwinglichen Aufwand und die Komplexität der manuellen Integration von Gegenmaßnahmen auf FPGAs zu adressieren, wird der AGEMA_FPGA EDA-Flow vorgeschlagen. Dieses Framework optimiert das Mapping bestimmter maskierter Gadgets erster Ordnung speziell für FPGA-Architekturen, indem es nicht-lineare Funktionen so zerlegt, dass sie an intrinsischen Einschränkungen von LUTs ausgerichtet sind, während die nachweislich sicheren Eigenschaften beibehalten werden.

Abschließend wird ein sicherheitsorientiertes Open-Source-Framework für rekonfigurierbare Architekturen nebst Toolchain präsentiert. Durch die Verwendung sicherer, kombinierbarer Gadgets und eines automatisierten Mapping-Flows ermöglicht das Framework die Implementierung von SCA- und FI-resistenten Schaltungen, ohne dass domänenspezifisches Fachwissen erforderlich ist. Dieser Ansatz erzielt eine Flächenreduzierung gegenüber herkömmlichen rekonfigurierbaren Architekturen, wobei die physische Sicherheit durch TVLA über 100 Millionen Traces validiert wurde. Zusammenfassend schlagen diese Beiträge eine Brücke zwischen theoretischen Hardware-Schwachstellen und praktischen, skalierbaren Verteidigungsmaßnahmen und schaffen so eine flexible und verifizierbare Grundlage für moderne Hardware-Roots-of-Trust.

Contents

List of own publications

Transactions and Articles

[1] S. Meschkov, D. R. E. Gnad, J. Krautter, and M. B. Tahoori, "New approaches of side-channel attacks based on chip testing methods", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2022

[2] K. Schoos, S. Meschkov, M. B. Tahoori, and D. R. Gnad, "Jitsca: Jitter-based side-channel analysis in picoscale resolution", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 3, pp. 294–320, 2023

[3] S. Meschkov, D. Lammers, M. B. Tahoori, and A. Moradi, "Design and implementation of a physically secure open-source fpga and toolchain", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2025, no. 3, pp. 542–582, 2025

Conferences

[4] S. Meschkov, D. R. Gnad, J. Krautter, and M. B. Tahoori, "Is your secure test infrastructure secure enough?: Attacks based on delay test patterns using transient behavior analysis", in *2021 IEEE International Test Conference (ITC)*, IEEE, 2021, pp. 334–338

[5] N. Muller, S. Meschkov, D. R. Gnad, M. B. Tahoori, and A. Moradi, "Automated masking of fpga-mapped designs", in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, IEEE, 2023, pp. 79–85

List of other publications not included in this thesis

[6] H. Xu, S. Meschkov, V. Meyers, and M. Tahoori, "Pwnn: Power-wasting neural network as remote fault injector", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2026, no. 1, pp. 448–471, 2026

[7] B. Sapui, J. Krautter, M. Mayahinia, A. Jafari, D. Gnad, S. Meschkov, and M. B. Tahoori, "Power side-channel attacks and countermeasures on computation-in-memory architectures and technologies", in *2023 IEEE European Test Symposium (ETS)*, IEEE, 2023, pp. 1–6

[8] B. Sapui, S. Meschkov, and M. B. Tahoori, "Side-channel attack with fault analysis on memristor-based computation-in-memory", in *IEEE 30th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, 2024

[9] S. M. Ghasemi, S. Meschkov, J. Krautter, D. R. Gnad, and M. B. Tahoori, “Enabling in-field parametric testing for risc-v cores”, in *2023 IEEE International Test Conference (ITC)*, IEEE, 2023, pp. 367–376

[10] S. M. Ghasemi, S. Meschkov, J. Krautter, D. R. Gnad, and M. B. Tahoori, “Slm isa and hardware extensions for risc-v processors”, in *2023 IEEE 29th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, 2023, pp. 1–5

[11] S. M. Ghasemi, J. Krautter, T. Gheshlaghi, S. Meschkov, D. R. Gnad, and M. B. Tahoori, “Degradation monitoring through software-controlled on-chip sensors for risc-v”, in *2024 IEEE European Test Symposium (ETS)*, IEEE, 2024, pp. 1–6

[12] S. M. Ghasemi, S. Meschkov, J. Krautter, D. R. Gnad, and M. B. Tahoori, “In-field detection of small delay defects and runtime degradation using on-chip sensors”, in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2024, pp. 1–2

Part I.

Introduction and Backgrounds

1 Introduction

In today's rapidly expanding digital world, many aspects of daily life rely heavily on information technology infrastructure, making security concerns greater than ever. Everyday activities such as online shopping and banking, payments with credit or debit cards, accessing buildings or cars with electronic keys, or communicating through instant messaging services, wireless networks, and mobile networks all rely fundamentally on cryptographic mechanisms. Beyond personal use, critical infrastructures, including electricity grids, healthcare, and public transportation systems, are becoming increasingly digitalized, which further amplifies the potential impact of security breaches. Consequently, ensuring that data, communication, and computing infrastructures remain resilient against malicious activity has become a matter of paramount importance. At the heart of these protections lies cryptography, which enables core security properties such as confidentiality, integrity, and authenticity.

Confidentiality safeguards sensitive information by ensuring that it can be accessed only by authorized parties, thereby protecting personal data, intellectual property, and classified communications from disclosure. Integrity guarantees that data remains unaltered during transmission or storage, preventing undetected tampering or corruption. Authenticity, in turn, establishes trust in the identity of communicating entities, ensuring that information originates from a legitimate source and not from an impersonator. Together, these principles provide the foundation of trust in modern computing systems.

Although these systems include both hardware and software components, the primary focus has traditionally been on software security, which has also been the subject of research for a longer period. This is largely due to the common assumption that hardware provides a secure root-of-trust, whereas the majority of security attacks are directed at software. However, as software security improves and compromising hardware becomes more studied, hardware security poses a significant vulnerability to the whole system. Evidence of hardware security breaches emphasizes the growing importance of hardware security in the academic, industry, and government sectors [13]–[15].

Physical attacks, such as SCA attacks [16]–[18], and FI attacks [19], [20] pose a significant challenge in security-enabled applications, especially for those operating in hostile environments. This is especially critical since compromising security-sensitive components such as the Trusted Platform Module (TPM) could result in a breach affecting any software running on the system. To grant some level of security of the system as a whole, both software and hardware need to be protected. Although it is impossible to fully prevent any SCA leakage, suitable countermeasures can make SCA attacks practically infeasible

or allow detection of fault injection attempts. Such countermeasures usually incorporate masking sensitive information and hiding techniques.

In this PhD thesis, the focus lies on both hardware attacks and corresponding defense strategies. Together with other researchers, we have recently highlighted the increasing severity of side-channel and fault attacks [1], [2], [4], [7], [8], some of which can even be executed remotely and without the need for expensive equipment. To counter such threats, we have elaborated flexible defense strategies based on masking and hiding techniques tailored for use on reconfigurable hardware [3], [5]. Central aspects here are: (1) the accessibility of such defenses for hardware designers who may not possess specific expertise in hardware security; (2) deployment of the hardware design on reconfigurable hardware for flexibility, allowing a certain level of adjustments against newly discovered threats. The first challenge is addressed through an automated workflow for an end-user, thereby enabling the straightforward application of security mechanisms. For the second challenge, we analyze and elaborate the use of commercially available FPGAs as well as develop a custom secure reconfigurable system tailored for protection against SCA and FI attacks at the gate level.

1.1 Contributions

1.1.1 Test-Infrastructure–Induced Side-Channel Attacks

Modern test-security practice is built on the premise that secrecy is preserved by restricting or encrypting access to sensitive parts of the Design-for-Testability (DfT) infrastructure, such as scan chains. In addition, the information available at publicly accessible outputs is assumed to be non-sensitive and poses no security risks. Consequently, tests performed on these publicly available, non-sensitive outputs are traditionally considered secure and assumed not to leak secret information. Furthermore, for achieving functional safety requirements, the on-chip test infrastructure is reused in-field and cannot be completely disabled after the manufacturing test phase. However, we show that having access to (small delay) test results on insensitive (public) outputs can in fact reveal secret data.

In Chapter 3 we formalize a realistic threat model in which the adversary is limited to observe signal values at publicly accessible outputs alongside with the according test data that reveals transient behavior of those public signals. This includes data collected during execution of timing tests such as (small) delay tests, path-delay, and speed-grading (Fmax) campaigns and does not require any privileged access into internal states. In the proposed attack scenarios, the adversary observes (1) full or down-sampled transient timing traces, (2) a few (strategically chosen for the test) time samples, (3) compacted responses such as MISR signatures, and, in the extreme, (4) binary pass/fail outcomes, retrieved from the test data. The resulting information is similar to insights retrieved in some FI attacks relying on timing faults, such as FSA[21]. We, however, treat the test data rather as side-channel observations and mount template-based attacks on the results of delay testing on the output of cryptographic circuits using real hardware. Templates are built on a reference

device and can be applied to a copy of this device. The attack requires only data from a small number of random test patterns on the victim. The methodology is robust to realistic nuisances: runtime variation, environmental noise, instrumentation inaccuracies, and down-sampling of measurement data.

The empirical results challenge the security assumptions that test outcomes of non-sensitive and publicly accessible signals are harmless. They further imply that timing-based tests must be evaluated for their information-leakage properties. Concretely, our results motivate (1) revising DfT infrastructure threat models to account for test result side-channels, (2) constraining the granularity and retention of delay data in production databases, (3) designing test patterns and compaction strategies that minimize correlation with secret-dependent dynamics, and (4) potentially enforcing to compute only with dummy data during test to avoid leakage of any secret. In safety-critical deployments, where in-field tests are indispensable, these measures are central to reconciling diagnosability with confidentiality.

1.1.2 Jitter-Only Side Channel over Communication Links (JitSCA)

In the context of safety- and security-critical systems, securing boundaries against physical side-channel leakages remains an ongoing challenge. A fundamental assumption in secure hardware design has been that strict segregation of power domains and the implementation of galvanically isolated communication vectors sufficiently mitigate power and electromagnetic side-channel vulnerabilities. However, recent empirical analyses necessitate a reevaluation of this premise.

The work presented in Chapter 4 identifies and characterizes a novel class of side-channel vulnerabilities, termed Jitter-based Side-Channel Analysis (JitSCA), demonstrating that galvanically isolated communication channels inherently propagate inadvertent data-dependent side-channel leakage through minuscule timing variations, formally quantified as signal jitter.

The investigation utilizes a high-resolution Time-to-Digital Converter (TDC) instantiated via configurable delay lines on FPGAs. This sensor architecture achieves a discrete temporal resolution of approximately 11.5 picoseconds. To systematically isolate the jitter-based leakage vector from conventional Power Distribution Network (PDN) coupling, the experimental design spans three progressively decoupled adversarial typologies:

1. **On-chip paradigm:** A baseline configuration sharing a monolithic PDN.
2. **Inter-device paradigm:** A cross-FPGA setup communicating via twisted-pair cabling, sharing a common ground potential but independent power supplies.
3. **Isolated inter-device paradigm:** A tightly controlled environment establishing communication via a differential Non-return-to-zero space (NRZS) HDMI link equipped with hardware galvanic isolation. In this definitive configuration, the cryptographic transmitter and the receiver operate from independent battery sources, with the transmitter localized within an electromagnetic-isolated thermal chamber.

The analytical framework relies on CPA[22] targeting a hardware implementation of the Advanced Encryption Standard (AES) core. Additionally, structural leakage is independently verified via TVLA[23] using a generalized higher criticism test statistic. To unequivocally prove that the source of the extracted cryptographic material stems from the temporal perturbations of the transmitted signal rather than induced voltage differentials on the attacker's delay line, a Phase Locked Loop (PLL) is introduced defensively at the receiver to filter incident signal jitter as a negative control variable.

The empirical evaluation yields decisive evidence that deterministic transient voltage fluctuations occurring during cryptographic execution impart statistically significant jitter onto outgoing digital signals, irrespective of direct electrical connectivity.

- Within the maximal isolation scenario (galvanically isolated HDMI), the data trace exhibited measurable signal jitter within a bounded margin of 54 ± 45 ps.
- The CPA profile successfully yielded a full key recovery from the AES accelerator. While the non-isolated on-chip baseline necessitated approximately 27,000 traces for key recovery, the cross-device attack through a galvanically isolated barrier reached convergence using as few as 47,000 traces.
- Galvanic isolation yielded a minimal security margin increase (a multiplier of merely $1.7\times$), demonstrating that inductive, differential isolation fails to attenuate temporal side-channel leakage.
- The application of the PLL filter efficiently suppressed the CPA correlation and minimized SNR, firmly excluding the hypothesis that the extracted data propagated through common-mode voltage fluctuations rather than signal timing characteristics.

These findings present profound implications for secure system architecture, conclusively disproving the axiom that galvanic separation categorically circumvents inter-device power side channels. By establishing that signal jitter effectively acts as a high-fidelity proxy for power variability, JitSCA exposes a pervasive threat vector. This vulnerability likely transverses diverse digital communication protocols characterized by contiguous electrical or optical synchronization, including USB, interconnect architectures (e.g., PCIe), and potentially fiber-optic networking.

To mitigate this subclass of timing side channels, traditional security paradigms must be re-architected. Current implementations of physical decoupling or logical unidirectional gateways (data diodes) are insufficient if they continuously propagate the deterministic temporal characteristics of internal cryptographic state transitions. Effective remediation will necessitate complex structural interventions at the physical and link layers, such as the integration of asynchronous packet-switched buffering topologies or deeply segregated FIFO queues spanning distinct decoupled clock and power domains to purposefully destroy the correlation between signal jitter and computational state.

1.1.3 AGEMA_FPGA: Automated First-Order Masking for FPGAs

The mitigation of power analysis vulnerabilities in FPGAs requires physically secure masking implementations. While composable masking schemes provide formal security guarantees under generic adversary models, their naive instantiation on reconfigurable hardware typically incurs prohibitive area overhead, degrades operational frequency, and demands excessive randomness. Consequently, automated generation of masked circuits has emerged as a critical requirement to circumvent the error-prone nature of manual hardware security design. However, existing automated frameworks, such as AGEMA[24], are intrinsically optimized for Application Specific Integrated Circuit (ASIC) standard-cell libraries. The direct translation of these ASIC-oriented netlists to FPGAs results in highly inefficient LUT utilization.

To resolve this platform-specific inefficiency, the research presented in chapter Chapter 5 introduces the AGEMA_FPGA EDA flow, which programmatically maps arbitrary unprotected FPGA netlists to provably secure, masked equivalents. Methodologically, the framework algorithmically decomposes non-linear target functions into sub-functions constrained to a maximum of five inputs. This structural partitioning enables the efficient point-wise application of GHPC[25] directly at the intrinsic FPGA LUT boundary. Concurrently, the EDA flow executes an algebraic parsing phase to explicitly distinguish linear Boolean functions and unmasked multiplexer primitives embedded within the LUT configurations. Rather than applying high-overhead non-linear masking, these linear and multiplexing elements are securely instantiated as independent geometric duplicates across distinct share domains. This bipartite combinatorial mapping deterministically minimizes spatial overhead by preventing the unwarranted deployment of cascaded masking structures.

The application of this architecture-aware synthesis yields substantial empirical improvements when benchmarked against unmodified ASIC-centric masking translations. Across comprehensive cryptographic case studies (including AES, CRAFT, Midori, Skinny, and LED), the FPGA-optimized primitives demonstrate architectural reductions encompassing up to 64% fewer LUTs and 22% fewer flip-flops, while simultaneously attenuating total power consumption by up to 59%. Furthermore, the EDA flow inherently sustains Probe-Isolating Non-Interference (PINI)[26], guaranteeing composable physical resilience under the robust glitch- and transition-extended probing models. The theoretical physical security properties were empirically validated via a standard fixed-versus-random TVLA[23]. The statistical evaluation across 100 million high-resolution power traces on a Kintex-7 target yielded a maximum t-statistic strictly below 4.5 ($|t| < 4.5$), confirming the absence of detectable first-order leakage.

The theoretical implications of these findings establish that formally verified masking paradigms can be efficiently implemented with reconfigurable logic hierarchies. By aligning composable side-channel protections directly with the underlying algorithmic LUT and multiplexer geometries, it is possible to achieve provable physical security without conceding fundamental resource efficiency. This formalization provides a deterministic trajectory for integrating robust hardware countermeasures into standardized FPGA synthesis

pipelines, thus abating the dependency on domain-specific cryptographic engineering expertise.

1.1.4 Security-First Open Reconfigurable Fabric and Mapping Flow

The integration of cryptographic primitives often relies on dedicated System on Chip (SoC) components or ASICs engineered to process sensitive data and execute cryptographic algorithms securely. While these highly customized hardware solutions offer robust protection tailored to specific functions, their intrinsic inflexibility makes adaptation to rapidly evolving security threats fundamentally challenging. Unlike software, which routinely receives updates to mitigate emerging vulnerabilities, conventional hardware remains immutable throughout its operational life cycle. Consequently, in some scenarios secure hardware solutions can become outdated very quickly.

The hardware inflexibility issue can be solved with reconfigurable fabrics, e.g., FPGAs, that play an important role due to enabling rapid time-to-market by flexibility and updateability. FPGAs are often used for security applications and allow hardware "security patches" similar to software applications. Nevertheless, commercial FPGAs are not specialized for security applications and many security issues still need to be resolved [27]–[31].

Integrating known countermeasures to physical attacks in conventional FPGAs is challenging and incurs significant penalties compared to ASIC solutions, resulting in higher area overhead, lower throughput, and increased latency. Implementation and mapping of such security schemes is usually re-done manually for every cryptographic algorithm and fabric architecture, which requires a high level of hardware security proficiency and development effort. In addition, developing designs dedicated to an FPGA platform is limited by the vendor tools and the primitives available on the device. Therefore, countermeasures might not provide the desired security level or lead to a high inefficiency with respect to performance.

For instance, there is a body of work focused on hiding countermeasures for conventional FPGAs, with some efforts exploring improvements through balanced placement and routing techniques [32]–[35], while others propose adjustments to the FPGA architecture or modifications to the implementation of LUTs [36]–[38]. However, these approaches are still not easily applicable to most commercially available FPGAs due to the aforementioned limitations. Moreover, since they rely solely on hiding countermeasures, the provided side-channel resistance is not verifiable through a formal security model and can be considered as insufficient.

In chapter Chapter 6 we propose a solution to address the challenges elaborated above, which serves as a roadmap to build a foundation enabling efficient realization of reconfigurable SCA and FI-secure devices. The main aspects of our contribution are summarized as follows:

Secure Reconfigurable Fabric Design Methodology. We propose an approach to

designing reconfigurable fabrics and a respective toolchain tailored with hardware security in mind. The concept provides resistance at the hardware level to various malicious physical attacks, such as SCA and FI attacks. A fabric designed by our methodology can act as the root-of-trust in SoCs in order to implement and execute cryptographic algorithms or functions demanding high security. Our technique preserves physical security at the gate level and is based on provable secure gadgets, subcircuits which are composable without violating security assumptions of the resulting circuit. Generally, many known gadget-based hardware security schemes can be utilized in the design process for such a fabric.

Fully Automated HDL Design Flow for Bitstream Generation. In fact, the mapping of a netlist to the fabric is crucial to maintaining security properties. If security assumptions are violated, physical security cannot be maintained. However, we made a customized toolchain to automate mapping as well as place-and-route for specific fabric architectures. As a result, no expertise is required in the hardware security domain, and insecure Hardware Description Language (HDL) designs are automatically mapped to achieve physically secure operation.

Entirely Open-Source Toolchain. We utilize AGEMA [24], FABulous [39], Yosys [40], nextpnr [41], and BitMan [42], [43]. These tools pave the way for efficient and customizable open-source development without vendor limitations. Our goal is to enhance security through comprehensive evaluation and expansion within the broader research community.

Secure Gadget-based Fabric Implementation. As a case study, we implemented a first-order SCA-secure reconfigurable fabric based on WDDL [44] and LMDPL [45] gadgets with the assistance of the above-mentioned open-source tools. The gadgets are adjusted to enable fault detection on the dual-rail signals according to the concept of Totally Self-Checking Circuits [46]. In this study, we present the complete design flow for secure fabric design. In particular, we explain the design of gadgets and how they are assembled into a reconfigurable fabric. Furthermore, we show the result of fully automated bitstream generation flow for several cryptographic algorithms given their unprotected HDL designs. Moreover, we conduct an area improvement estimation. Compared to a conventional FPGA architecture (FABulous reference implementation), our mapped HDL designs result in reduced area consumption by approximately 85%. TVLA on real power measurements of an emulated FPGA platform confirms no leakage detection over 100 million traces.

By shifting the burden of physical security to the fundamental platform infrastructure, this framework establishes a flexible and verifiable hardware root-of-trust.

1.2 Outline

The thesis is organized into the following chapters:

- Chapter 2 summarizes the essential background information required for this thesis, such as Side-Channel Analysis fundamental, leakage assessment, and core countermeasures as well as fault attacks.
- Chapter 3 raises attention to DfT-infrastructure and test data by showing that small-delay/Fmax test results on “public” outputs leak secrets.
- Chapter 4 introduces JitSCA: jitter-based SCA which demonstrates leakage and key recovery across on-chip, cable-coupled, and even galvanically isolated links; separates clock-jitter from voltage effects and highlights that tiny timing variations on communication channels can be exploitable.
- Chapter 5 presents AGEMA_FPGA, an automated flow that maps GHPC-style masked gadgets efficiently onto FPGA fabrics resulting in provably secure first-order masked circuits.
- Chapter 6 proposes a security-first open reconfigurable fabric based on masked gadgets and open mapping flow.

2 Backgrounds

In this chapter, we provide the necessary background to understand the problem of fault and side-channel attacks together with their corresponding countermeasures. We begin by introducing the concept of Side-Channel Analysis (SCA) and describing the basic attack techniques relevant to this thesis, such as CPA and template attacks. Next, we discuss leakage assessment methods, which are essential for evaluating the effectiveness of countermeasures, and provide an overview of countermeasure strategies against SCA, including the specific implementations considered in this work. Finally, we introduce the class of Fault Injection (FI) attacks and outline common countermeasures designed to mitigate them.

2.1 Side-Channel Analysis (SCA)

SCA attacks have shown that even if the underlying security algorithm is mathematically proven to be secure, the insights gained from the observable physical effects of its implementation during the processing of sensitive data can be used by an attacker to compromise its security. Such observable side channels are, for example, power consumption [17], electromagnetic emanations [18], runtime behavior [16] and many others.

These attacks often target devices to which an attacker can gain physical access, for example, to measure their power consumption with an oscilloscope. However, some power analysis and electromagnetic attacks do not require dedicated measurement equipment and have been shown to be feasible from within the same chip [30], [47], the same power domain [48], or in close proximity [49]–[51]. This enables some remote SCA attacks, such as the attack exploiting signal jitter in a communication link [2]. Especially, multi-user FPGA environments in the cloud are seen as a well-suited target for such remote SCA attacks, since a malicious FPGA user can implement different types of power sensors, such as TDC [28], Ring Oscillator (RO) [52] or routing based [53], to attack other users or components in the same PDN.

Certain physical defaults are nowadays well known to cause or amplify SCA leakage. This includes glitches [54], transitions [55], and coupling effects [56] during normal operation of the device. In the context of this work, we conduct the security analysis on gate level. Hence, we do not discuss couplings, but cover glitches and transitions.

Glitches. A glitch is a transient physical effect associated with unintended switching

activity in Complementary Metal Oxide Semiconductor (CMOS) circuits due to different timings in signal propagation [57]. In each clock cycle, transient changes due to glitches cause significant dynamic power consumption until the internal state and output of a circuit reach a stable state, i.e., ready to be stored in registers. This behavior (pattern of glitches) depends on the processed (secret) data and may cause or amplify exploitable leakage.

Transitions. When the result of a computation overwrites a stored value in a register, the associated power consumption is dependent on both values. This potentially results in secret-dependent power consumption and is referred to as transitional leakage in the literature.

2.1.1 Correlation Power Analysis (CPA)

Correlation Power Analysis (CPA)[22] is one of the most widely used statistical techniques in side-channel cryptanalysis. The central idea is to exploit the fact that the instantaneous power consumption (or electromagnetic emanation) of a cryptographic device often correlates with intermediate values of the algorithm being processed. By comparing measured leakage traces against predictions derived from a leakage model, an adversary can identify the correct secret key.

To reveal its statistical power, first of all, CPA requires a large amount (up to hundreds of thousands) of side-channel measurements while the device encrypts (or decrypts) varying data. It is generally assumed that the attacker has access to the ciphertext – after all, there would be no need for encryption if the communication channel is fully secured. A leakage model is then chosen to approximate how data values influence power consumption. Common models include the Hamming Weight (HW) - number of '1' bits in a register value, and the Hamming Distance (HD) - number of bit transitions between two states. For each key hypothesis key_{guess}, the attacker computes the hypothetical intermediate value (such as the output of an S-box) and calculates its predicted leakage according to the chosen model. This results in a hypothetical power consumption vector for every key guess. In the final step, Pearson's correlation coefficient ρ is computed between each hypothetical power consumption vector and real side-channel measurements as follows:

$$\rho = \frac{\operatorname{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{2.1}$$

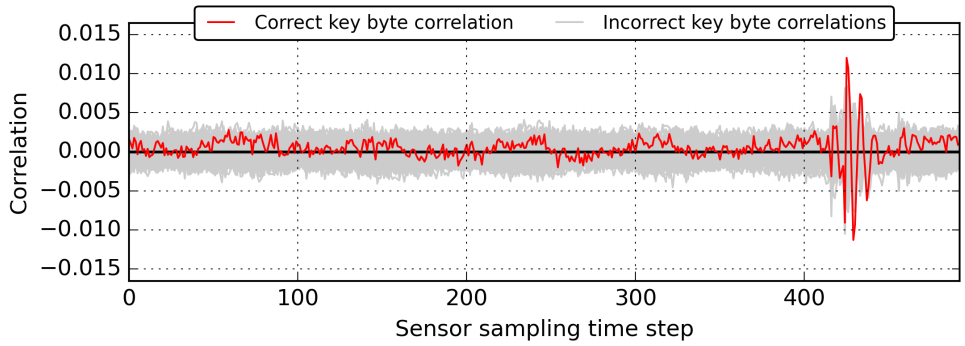
where X is the hypothetical leakage vector for a key guess and Y represents the measured traces at a given sample point. A correlation close to zero indicates little relation, whereas a large absolute value indicates that the hypothesis aligns with the measured leakage. By scanning across all time samples, the correct key guess typically produces a sharp correlation peak, while incorrect guesses produce noise-like correlations.

In practice, CPA attacks against block ciphers such as AES are performed using a divide-and-conquer strategy. Instead of attempting to recover the entire secret key at once, the attacker exploits the algorithm's structure to target intermediate values that depend on small, independent subkeys. In the case of AES, for example, after the SubBytes operation in the final round, each output byte is combined with one key byte through the AddRoundKey step, without further diffusion by MixColumns. This independence allows the attacker to focus on one output byte at a time. For each key-byte hypothesis, the corresponding state byte is computed (e.g., by inverting the S-box using known ciphertexts), and its hypothetical leakage is estimated under a model such as HW. The full key can be recovered in the following way:

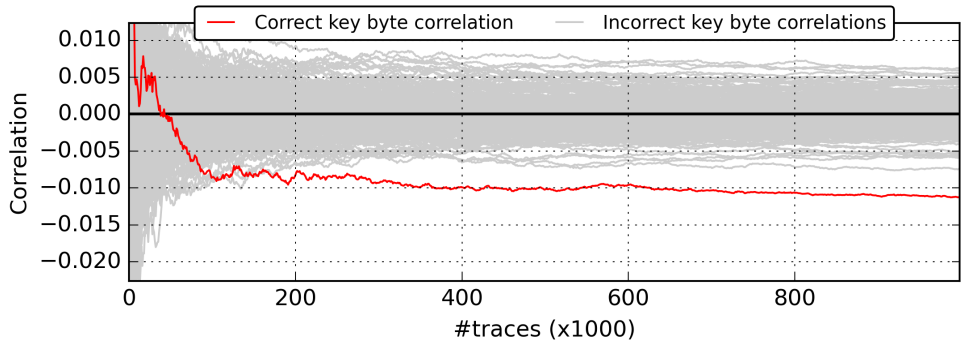
1. Select first key byte.
2. For each $keybyte_{guess} \subset \{0, 1, \ldots, 255\}$ compute the power model $P_{mod}(keybyte_{guess})$ which approximates the power consumption of a key-dependent operation in the encryption.
3. For each time point in the side-channel measurement t, compute the Pearson correlation coefficient ρ of $P_{mod}(keybyte_{guess})$ and over all measurement traces at time point t.
4. Repeat previous steps for all remaining subkeys.
5. For each subkey, identify which $keybyte_{guess}$ has a significantly higher absolute correlation value than all other key guesses and select it as part of the full secret encryption key.

This way, an attacker can, for example, instead of guessing 2^{128} keys, reduce the problem to 16×2^8 subkeys. In Figure 2.1 we provide an illustrative example of a CPA attack targeting a single AES key byte. The figure highlights two complementary perspectives on correlation results. For both plots, the correlation coefficients for incorrect key guesses are colored grey, whereas the correlation with the correct secret key byte is marked red.

Figure 2.1(a) presents the correlation coefficients for all key hypotheses across the complete sampling period after a fixed number of traces has been collected. All correlation values and especially the correct key hypothesis, highlighted in red, produce a distinct correlation peak around sample point 425, while being significantly closer to zero at other time steps. In contrast, Figure 2.1(b) shows how the correlation value evolves at this specific sample point as more traces are added. This progression clearly indicates the point at which the attack becomes successful: in this example, a reliable key recovery is achieved after approximately 200,000 traces. Together, these plots demonstrate both how leakage can be localized in time and how the number of required traces for successful key extraction can be determined.



(a) Total correlation values over the entire side-channel measurement.



(b) Correlation values over the amount of evaluated traces in one specific sampling time point.

Figure 2.1.: Illustration of a Correlation Power Analysis (CPA) attack on a single AES key byte: correlation values across the entire sampling period, and evolution of the correlation coefficient at a specific time sample as additional traces are collected.

2.1.2 Template Attack

Template attacks [58] are a special kind of side-channel attacks which are often used when no suitable leakage model is known. This attack type consists of two different phases, the first one is the *templating* or *profiling* phase and the second one is the template matching or attack phase. In the first phase, the attacker has full access to an identical copy of the *victim device* and collects a set of traces with all known inputs and outputs to obtain device characteristics. Where an identical device is unavailable, a simulation can also be sufficient for the attacker to perform the required measurements.

In the second phase, the characterization of the device together with collected side-channel information from the *victim device* is used to determine the secret. In practice, the attack phase can be used in different ways. One option is to remove noise from side-channel information, and the second option, which we use in this work, is to make a prediction on the data-dependency of the *victim device*.

Often, it is hard to find a precise model based on a mathematical function to predict the behavior of the *victim device*. Therefore, an empirical model built during the templating

phase is an easier way, which typically allows a much faster recovery of the secret compared to the use of a mathematical function as the model. In this thesis, we generally use Pearson correlation for the statistical template matching phase, whereas other methods are also shown feasible in the literature [58].

2.2 Leakage Assessment

It is often desirable to evaluate an implementation in order to determine whether it is vulnerable to side-channel attacks. However, given the continuous emergence of new attack techniques, it is impractical to verify the security of an implementation against all possible threats individually. Leakage assessment addresses this challenge by analyzing power traces of the Device Under Test (DUT) and determining whether any data-dependent information can be observed. Such a practical, attack-independent methodology that uses statistical hypothesis testing (Welch's t-test) to decide whether side-channel measurements from a DUT contain exploitable leakage is Test Vector Leakage Assessment (TVLA) [23].

2.2.1 Test Vector Leakage Assessment (TVLA)

A t-test evaluates whether two sets of data are significantly different by testing the null hypothesis that they originate from populations with the same mean. Welch's t-test is a commonly used variant in leakage assessment because it does not assume equal variances between the two groups. It computes a t-statistic from the difference of sample means normalized by an estimate of the pooled variance, and then derives a probability value that quantifies how compatible the observed data are with the null hypothesis.

Welch's t-test is defined as follows, where μ_i is the sample mean, s_i^2 the sample variance, and n_i the sample size for group i:

$$\mathrm{t} = \frac{\mu_0 - \mu_1}{\sqrt{\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}}} \tag{2.2}$$

In practical side-channel TVLA [59], the resulting statistic is used as a detection metric, where large |t| (commonly a threshold $|\mathrm{t}| > 4.5$) is taken as evidence to reject the null hypothesis of "no difference" and therefore indicates leakage. It is important to highlight here that this approach reports detectable leakage but does not itself quantify how easy an attack would be.

TVLA can be executed as *specific* or *non-specific* (often called *fixed vs. random*) t-test. In a *specific* t-test the traces are partitioned according to a chosen intermediate value (e.g., a particular S-box output bit or byte) so the test directly targets a hypothesized leakage point and model; this makes the specific test powerful for identifying which intermediate values leak and is helpful when preparing a key-recovery attack. By contrast, the *fixed vs. random* t-test is model-independent: one preselects a fixed associated datum D and

randomly interleaves executions with that fixed D and with fresh random inputs, then compares the two groups. This detects general exploitable leakage without assuming a target or leakage model. Therefore, *fixed vs. random* t-test variant is the standard TVLA practice for broad leakage checks.

2.2.2 Signal to Noise Ratio (SNR)

Signal to Noise Ratio (SNR) is a well-established concept in electrical engineering and signal processing. In general terms, it is defined as

$$\mathrm{SNR} = \frac{\mathrm{Var}(\text{ signal })}{\mathrm{Var}(\text{ noise })} \tag{2.3}$$

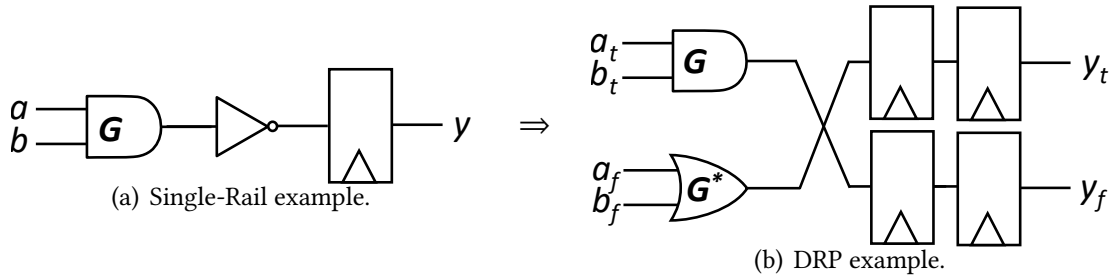
where Var( ) denotes the variance of a random variable. For a fixed time sample let X_t denote the signal component, which corresponds to the portion of the leakage relevant to the attack. The signal-to-noise ratio at time t is then defined as

$$\mathrm{SNR}_t = \frac{\mathrm{Var}\,(X_t)}{\mathrm{Var}\,(N_t)} \tag{2.4}$$

Var (X_t) measures the extent to which the leakage varies at time sample t due to the signal, while Var (N_t) quantifies the variation caused by noise. Thus, the signal-to-noise ratio expresses how much information is leaked at time sample t through the measurements. A higher SNR indicates a stronger data-dependent component in the leakage and therefore a greater susceptibility to side-channel attacks, whereas a lower SNR reflects a higher influence of noise and reduced observability of the signal. This, for instance, makes it possible to qualitatively compare different sources of leakage.

2.3 Hiding

A commonly known category of countermeasures is hiding, which attempts to decrease the SNR, so that the exploitable leakage is hidden below the noise level. This can be achieved by (1) noise amplification (addition of randomized switching activity or instructions) or (2) signal reduction (power equalization for all operations independent of processed data). Such concepts are well suited for hardware implementations. In particular, power-equalization schemes (also called Differential Power Analysis (DPA)-resistant logic styles) are primarily designed for ASIC platforms, since those are deployed at low abstraction levels, such as transistor to gate levels. Examples of such logic styles are Sense Amplifier Based Logic (SABL) [60], Wave Dynamic Differential Logic (WDDL) [44], Masked Dual-Rail Pre-charge Logic (MDPL) [61], and Dual-Rail Random Switching Logic (DRSL) [62], which predominantly rely on the Dual-Rail Pre-charge Logic (DRP) concept.



(a) Single-Rail example.

(b) DRP example.

Figure 2.2.: DRP example. a) depicts the conventional single-rail datapath: inputs a, b enter a logic block G, the result is inverted by a follow-up inverter gate, and the signal is stored in a register producing output y. b) shows the corresponding dual-rail encoding: each logical input is split into a true and false rail (a_t, a_f), (b_t, b_f) and processed by complementary logic blocks G and G^*, the output of gates is inverted by swapping the wires and the result is stored in two register stages for each rail, enabling alternation of the pre-charge and evaluation phases of the DRP protocol.

2.3.1 Dual-Rail Logic

In contrast to conventional single-rail logic, where a logical signal $x$is carried on a single wire, dual-rail logic uses two complementary wires, often called the true rail and the false rail, $(x_\mathrm{t}, x_\mathrm{f})$, where x_t is equivalent to the original signal and x_f to its complement. A logical value "1" is encoded as (1,0), while a logical "0" is encoded as (0,1). The state (0,0) is considered invalid (except in pre-charge logic, see below), and (1,1) is typically forbidden. Consequently, logic functions are replaced by dual-rail modules that process both rails in parallel and generate dual-rail outputs. In dual-rail (including DRP) circuits, no inverters are required and a logic signal is inverted simply by swapping the two complementary wires that carry this logic signal. Figure 2.2(b) shows how a simple single-rail logic path can be represented in a dual-rail implementation.

The structure with two complementary rails in dual-rail logic is supposed to balance out the loads during any operation in the circuit, resulting in a constant power profile. However, there are some effects which still cause side-channel leakage: (1) the early propagation effect [63], which may occur due to different input signal delays at a gate, and (2) the glitches, unintended switching activity propagating in the circuit due to early propagation effects, which significantly contribute to the power consumption of the circuit and may cause leakage in this way. To solve this, DRP schemes were introduced.

An additional issue to deal with is imbalanced routing [64]. Routes of different lengths and thus capacitive loads have different contributions to the amount of power consumption of signal toggle. Therefore, dual rails in a DRP scheme should use balanced routes to minimize the corresponding data-dependent leakage.

2.3.2 Dual-Rail Pre-charge Logic (DRP)

DRP logic styles consist of two alternating phases, namely the pre-charge and evaluation phases. The pre-charge phase sets every input and internal signal to a default state (usually

(0,0)), followed by the evaluation phase, where actual data is given to and processed by the circuit. Every gate behaves monotonically, i.e., in each phase every gate output toggles in one direction only. This way, the number of toggles in the circuit is constant in both phases and hence becomes data independent. A valid dual-rail encoding is defined as follows:

$$(x_t, x_f) = \begin{cases} (0,1) & \Rightarrow x = 0 \\ (1,0) & \Rightarrow x = 1 \\ (0,0) & \Rightarrow x \text{ in pre-charge phase} \end{cases}$$

A proper DRP circuit aims to eliminate data-dependent transition counts and glitches, ensuring exactly one pre-charge and one evaluation transition per cycle, since the evaluated results are always overwritten by the default state. Therefore, DRP is considered to provide stronger side-channel resistance compared to non-pre-charge dual-rail implementations. However, in due to process variations, DRP logic cannot provide perfect security against SCA attacks and only hardens the implementation against them.

2.3.3 Wave Dynamic Differential Logic (WDDL)

WDDL [44] is a well-studied DRP scheme that replaces gates with their dual-rail counterpart composed of standard cells. It targets an equal power consumption independent of the processed data. However, multiple follow-up works pointed out flaws in practice. First, it has been shown that its time-of-evaluation is data dependent [63]. Subsequently, a modified variant (DPL-noEE: "no early-evaluation") has been introduced for FPGAs in [65] to avoid its *early propagation* effect. However, the approach does not solve the same issue during the pre-charge phase. Another work introduced Asynchronous Wave Dynamic Differential Logic (AWDDL) [35] that fully avoids this issue also only for FPGAs.

In short, DRP as a standalone countermeasure does not prevent information leakage in practice. Slight routing imbalances and manufacturing variations lead to different capacitive loads that contribute to total power consumption. Therefore, dual rails should assure balanced routes to minimize (but not prevent) leakage. For example, the fat-wire approach [64] can be employed to minimize imbalanced delays in ASIC designs. However, the wire capacitance of the dual rails still differs slightly in the fabricated designs (due to process variations), potentially causing exploitable leakage. As a side note, the combination of DRP and a suitable masking scheme can provide a considerably high level of resistance in practice [66].

2.4 Masking

The concept behind Boolean Masking [67] is to randomize intermediate values of a cryptographic algorithm to make observable SCA leakages independent of predictable processed data. Based on secret sharing [68], d-th order Boolean Masking splits a sensitive message m

into $d + 1$ shares. The sharing consists of d individual and independent shares distributed uniformly at random. The last share x^d is computed as $x^d = (\bigoplus_{i=0}^{d-1} x^i) \oplus m$. Due to uniformity, an adversary who has access to at most d shares is unable to recover any information about m. Therefore, a $d + 1$ sharing is considered d-order secure. To maintain security during the entire computation, many masking schemes require fresh randomness in every clock cycle.

In this work, we aim for first-order security, i.e., a secret m is concealed with a random mask r, such that $x = x^0 \oplus x^1$, with $x^0 = r$ and $x^1 = m \oplus r$. Our modified masking schemes operate in DRP, such that fresh randomness must be provided every other clock cycle, i.e., updated for every evaluation phase.

2.4.1 Probing Models

In order to ease the evaluation and prove the security of masked implementations, some models have been defined to abstract the leakage behavior of the circuits. In the d-probing model [69] a formal adversary is defined, called the d-probing adversary, that is capable of observing any d circuit wires at a specified clock cycle simultaneously. According to this model, a circuit is considered d-probing secure if and only if a d-probing adversary is unable to recover any information about sensitive variables. As this approach relies on the observation of stable signals only, an extension, called robust probing model [70], also covers known physical defaults, namely couplings [56], transitions [55], and glitches [54]. In this model, a transition-extended probe adds a time dimension and records the value of the same wire in consecutive clock cycles. It is noteworthy to highlight that, due to the underlying DRP scheme employed in our methodology, transition-extended probes do not add any additional leakage/information. Hence, the security analysis in this work utilizes glitch-extended probes explained below.

Glitch-extended Probing Model. Since the number of glitches and their pattern depend on all changes occurring on the circuit's inputs, a glitch-extended probe placed on a wire propagates backwards to the last synchronization point (registers) or primary inputs of the circuit. By that, a formal adversary simultaneously probes all signals that contribute to the observed wires in the same clock cycle.

2.4.2 Gadget-based Masking and Composability

With increasing circuit complexity, especially in higher security orders, the efficient application of masking becomes increasingly difficult and error-prone. Unfortunately, designing a large provably secure circuit becomes infeasible quickly. A circuit composed of multiple individually-secure subcircuits may violate security assumptions under robust probing model. The probes placed on the primary outputs of a composed circuit are extended up to the last synchronization point, even across multiple subcircuits due to glitch probe propagation. Composability notions generally limit such probe propagations

to ensure that security in the model is preserved when the subcircuits are composed. A provably secure subcircuit that fulfills specific conditions to be composable without violation of security assumptions is commonly called a secure (composable) gadget. Such a gadget may implement a function as small as a simple two-input logical AND gate [71] or as large as an entire S-Box [25], [72]. Notable composable notions in the recent literature are Strong Non-Interference (SNI) [73], Glitch Strong Non-Interference (GSNI) [74], and PINI [26].

d-Strong Non-Interference (SNI). A gadget is considered d-SNI secure if and only if any p_1 probes placed on internal wires and p_2 probes placed on output wires, with $p_1 + p_2 \leq d$, reveal no secret information.

d-Glitch Strong Non-Interference (GSNI). GSNI is an extension of SNI by *glitch-extended* probes. Therefore, a gadget is considered d-GSNI secure if and only if any p_1 *glitch-extended* probes placed on internal wires and p_2 *glitch-extended* probes placed on output wires, with $p_1 + p_2 \leq d$, reveal no secret information.

d-Probe-Isolating Non-Interference (PINI). PINI is less restrictive than SNI by limiting the probe propagation to the probes' corresponding share domain. This allows the share-wise implementation of linear functions, e.g. XOR gates, without refreshing and leads to more efficient designs compared to SNI in practice. Formally, a gadget is d-probe-isolating non-interferent (d-PINI) if, for any set P of internal probes and any set $B \subset \{1, \ldots, n\}$ such that $|P| + |B| \leq d$, there exists $A \subset \{1, \ldots, n\}$, $|A| = |P|$ such that the probes in P and the output shares with index in B can be simulated with the input shares with index in $A \cup B$.

2.4.3 Generic Hardware Private Circuits (GHPC)

Contrary to Hardware Private Circuits (HPC) 1/2/3, GHPC [75] allows realizing an arbitrary vectorial Boolean function $\mathsf{F} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ as a single first-order secure gadget. Hence, gadgets are no longer restricted to atomic components, and even more complex components, e.g. S-boxes, can be securely composed. Independent of F, each GHPC gadget encompasses two consecutive register stages and needs a single fresh mask bit per output, i.e., m in total for F. Further, GHPCLL, a clock-cycle reduced variant of GHPC, results in gadgets with a single register stage. However, $m \times 2^n$ fresh masks are needed instead of m.

By denoting single-bit Boolean variables by $x, y, \ldots, z, q$ and their corresponding two shares by $x^0, y^0, \ldots, z^0, q^0$ and $x^1, y^1, \ldots, z^1, q^1$, formally GHPC transforms a Boolean function $\mathsf{f} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ with $\mathsf{f}(x^0 \oplus x^1, \ldots, z^0 \oplus z^1) = q^0 \oplus q^1$ by constructing 2^n Shannon cofactors, i.e. $\{\mathsf{f}|_{(0,\ldots,0)}, \ldots, \mathsf{f}|_{(1,\ldots,1)}\}$ such that $\mathsf{f}|_{(0,\ldots,1)} = \mathsf{f}(x^0, \ldots, \overline{z^0})$. We visualize the resulting hardware module in Figure 2.3. Note that the function output $\mathsf{f}(x, \ldots, z) = q$ is represented by two Boolean shares (q^0, q^1).

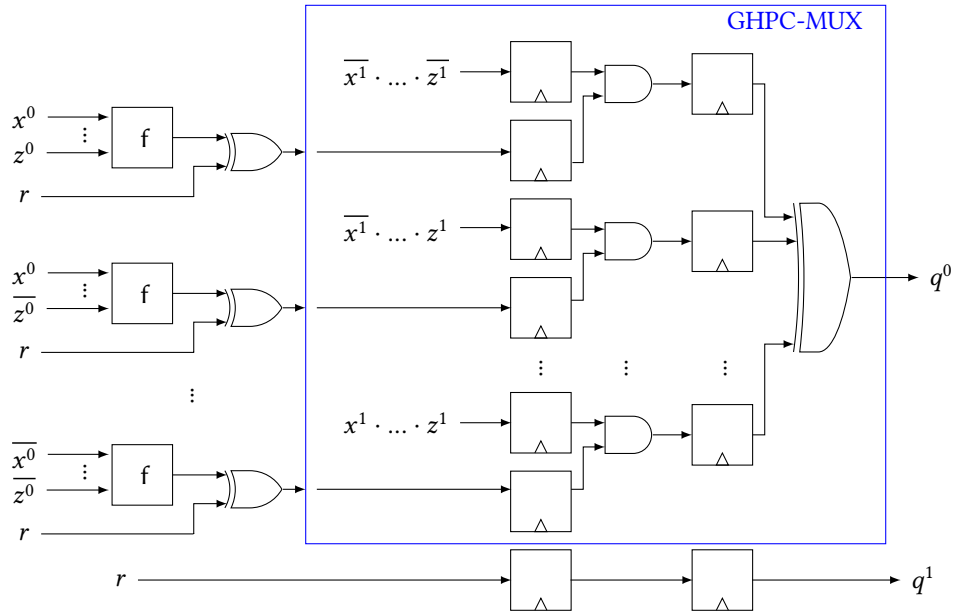


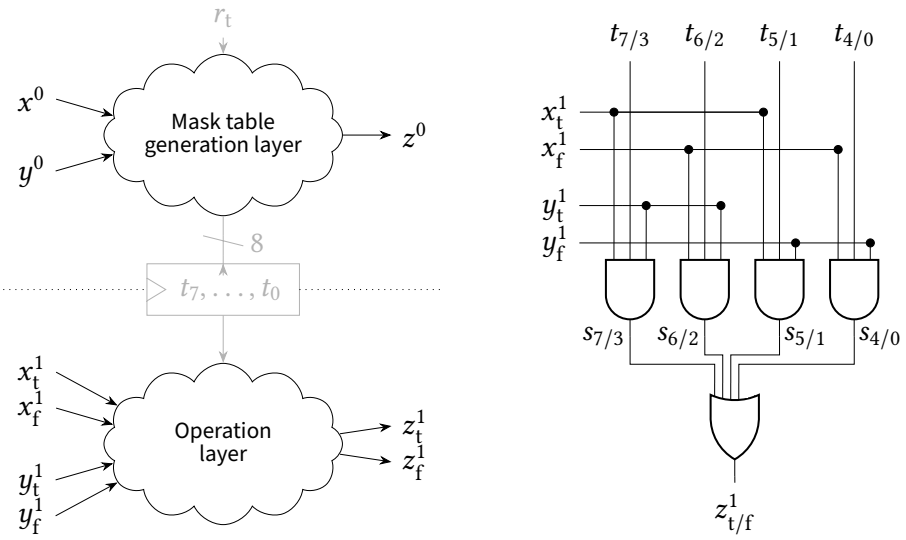
Figure 2.3.: Generic GHPC hardware module, partially taken from [75].

Initially, all Shannon cofactors are computed by forwarding the appropriate input to f. Next, the result gets refreshed by a single fresh mask r and stored in a register. The remaining logic (indicated by blue borders) acts as a clocked multiplexer forwarding the correctly blinded cofactor to the output.

2.4.4 LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL)

The gadget-based LMDPL masking scheme that combines DRP and masking to achieve first-order security has been originally introduced in [76]. The proposed scheme operates in a glitch-free manner, avoids the *early propagation* effect, and can be implemented on FPGAs or using ASIC standard cell libraries. In a follow-up work [45], the authors formally proved its security under the 1-GSNI composability notion and introduced an AES implementation that needs (on average) a single clock cycle per cipher round. The corresponding experimental evaluations confirmed first-order SCA resistance and even indicate a high level of resilience against higher-order attacks. Further third-party investigations confirmed its security [77].

Conceptually, the scheme is divided into the mask table generation and the operation layer as depicted in Figure 2.4(a). Considering a two-input non-linear (AND) gate with the inputs being x and y and output $z = xy$, the mask table generation layer processes the first shares x^0 and y^0 in their single-rail representation. It further requires a single fresh random bit r to blind intermediates stored in a register in between the layers while setting the first output share $z^0 = r$. The operation layer (as shown in Figure 2.4(b)) employs the DRP protocol and processes dual-rail shares $(x_\mathrm{t}^1, x_\mathrm{f}^1)$ and $(y_\mathrm{t}^1, y_\mathrm{f}^1)$ as well as the intermediates



(a) LMDPL layers overview.

(b) Non-linear LMDPL op. layer for true/false rails.

Figure 2.4.: LMDPL gadget structure.

given by the table generation layer to issue the second output share also in DRP form, i.e., $(z_t^1, z_f^1) = (xy \oplus r, \overline{xy} \oplus r)$.

2.4.5 Automated Generation of Masked Hardware (AGEMA)

The publicly available AGEMA framework [24] enables mapping of an arbitrary unprotected gate-level netlist of an ASIC design to a gadget-based d-order masked circuit. Therefore, even inexperienced hardware designers with limited time and budget are enabled to create provably secure masked hardware circuits while avoiding security flaws. AGEMA supports various securely composable gadgets such as HPC1/2/3 [71], [78], GHPC and GHPC_{LL} [25], and others. In addition, this tool is capable of applying the masking on selected parts that are annotated by the designer, i.e., sensitive parts of the circuit. Annotations indicate that secret data is processed, such that uncritical components are left unmasked, e.g., control logic, which can reduce the overheads significantly. Furthermore, AGEMA can optimize the results according to specific requirements in terms of area utilization, clock cycle count, and randomness.

An extension of the original AGEMA further supports efficient and automated mapping of the given gate-level netlists to masked circuits for FPGA platforms [5]. This extension is optimized for better usage of FPGA resources like LUTs and registers. For the same annotated given design and the same countermeasures, it achieves results with significantly less resource utilization and power consumption when mapped to an FPGA, compared to the original AGEMA framework.

Table 2.1.: Fault models at different levels.

Fault Level	Fault Model
Electrical and Logic level	Bit flip, Bit Set, Bit Reset Stuck-At Fault Delay Fault Transition Fault
Micro-architectural level	Register Corruption Memory Corruption Instruction Bus Corruption Bad fetch Pipeline and MAB Corruption
ISA level	Instruction Skip, Skip and Repeat Register Corruption Incorrect Instruction Execution Partial update
Software level	Control Flow Error Variable Corruption

2.5 Fault Injection (FI) Attacks

Contrary to passive observations in SCA, Fault Injection (FI) attacks [19], [20], [79] are considered as active attacks, in which the attacker actively attempts to make disturbances to the device with the goal of breaking its security properties. Such disturbances include for example voltage and clock manipulations, laser shots, electromagnetic pulses, or sudden temperature variations (see Table 2.2). A successful fault injection then results in corruption of the control or data path logic, or both (see Table 2.1). Subsequently, the attacker for instance can exploit the resulting incorrect outputs or corrupted internal states to extract secrets, bypass checks, or cause predictable failure modes. For example, as shown by Differential Fault Analysis (DFA) in [20], an attacker can analyze gathered information from the correct (non-faulty) and/or incorrect (faulty) outputs to recover some information about the secret. In general, the fault analysis techniques applied to recover secrets during a fault attack can be classified in three major categories (1) differential (e.g. [20], [80]), (2) collision based (e.g. [81]) and (3) statistical (e.g. [21]).

Fault attacks often require some knowledge about the device and/or implemented algorithm. However, in many scenarios it is also possible to bypass cryptographic security without requiring algorithmic knowledge, for example, adversaries may skip entire encryption routines by inducing faults in conditional branches[82]. Additionally, some fault attacks do not require actual faulty outputs for the analysis. For instance Ineffective Fault Analysis (IFA)[83] and specifically Statistical Ineffective Fault Analysis (SIFA)[84] exploit ineffective faults, which means correct, non-faulty ciphertexts produced despite a fault

Table 2.2.: Summary of Fault Injection Methods and Requirements

Method	Requirements	Example of necessary equipment	Cost	Temporal precision	Spacial precision
Clock glitching	Access to internal clock, generation and introduction of different clock waveforms.	- Oscilloscope - Clock Fault Generator	Low	High	Low
Voltage glitching	Access to efficient ways of effecting the power supply.	- Oscilloscope - Voltage Fault Generator	Low	High	Low
Temperature Variation	Changing the ambient temperature	- Temperature controlled chamber - Temperature sensors	Medium	Low	N/A
Electromagnetic	Electromagnetic pulse shape generation at desired location on chip.	- Electromagnetic probes - Probe positioning - Pulse Generator - Oscilloscope	Medium	High	Medium
Laser	Chip decapsulation and high precision laser spot generation.	- XYZ Table - Oscilloscope - Laser Control - Laser Source	High	High	High

injection attempt. By identifying when an injected fault (like a stuck-at fault) has no effect on the output, an attacker can determine specific intermediate state values.

Traditionally fault attacks require access to the target device and special hardware to enable the ability of injecting faults[81]. However, some attacks can still be executed remotely by exploiting unintended use of shared system components. For example, in [29] the proposed attack utilizes ROs to induce voltage drops in the shared PDN of a multi-user FPGA, which results in timing faults. Another work targets a Central Processing Unit (CPU) from an Embedded FPGA (eFPGA) in an SoC system with a power-hammering circuit [85].

2.5.1 Fault Sensitivity Analysis (FSA)

The specific delays of activated paths in a circuit are data dependent and can be used to reveal the secret information. Fault Sensitivity Analysis (FSA) [86] makes use of this knowledge and is based on the idea of increasing the intensity of the fault injection gradually until some distinguishable characteristics can be detected at the output.

In practice, this means that the power supply or external clock is disturbed in such a way that an internal clock cycle becomes shorter. With every iteration, the fault intensity is

increased, and the clock cycle gets shorter, until some faults begin to occur or become stable at the output. The knowledge about the fault intensity at which these faults are occurring or becoming stable is then called fault sensitivity and can be used as side-channel information by an attacker.

If the data dependency of the underlying system is known, a prediction of fault sensitivity based on inputs can be made; otherwise, a profiling phase of the template attack can be used to learn it [87], [88]. Finally, the fault sensitivity values observed from the *victim device* can be compared with the profile or prediction, e.g. via Pearson correlation. The best matching values (highest correlation coefficient) should correspond to the secret.

2.5.2 Countermeasures against Fault Injection Attacks

The mitigation of Fault Injection Attacks is a significant challenge in hardware security, requiring a defense-in-depth approach that spans the architectural, logic, and software layers. Current research focuses on two primary domains of implementation: hardware-level and software-level countermeasures, each characterized by distinct trade-offs between security granularity and resource overhead. Some of these are categorized as follows:

- **Prevent Fault Injection Attempts**
  - Active: Monitors or Sensors (e.g. Configurable Delay Block (CDB)[89])
  - Passive: Special Encapsulations
    - Tamper-Resistant/Active Shielding Packaging
    - Electromagnetic Shielding
    - Light-Blocking/Optical Packaging
    - ...
- **Mitigate Fault Effect**
  - Redundancy (spatial/temporal, full/partial)
    - Error Detection (e.g. WDDL[44])
    - Infection of faulty outputs (e.g. [90])
    - Error Correction (e.g. Error Correcting Code [91])
    - Prevention (e.g. Execution Randomization [92])
  - Protocol
    - Re-keying [93]
    - Tweak-in-Plaintext [94]
    - ...

- Cipher Level Technique
  - Cipher DEFAULT [95]

Part II.

Contents

3 Side-Channel Attacks based on Chip Testing Methods

The work described in this chapter was first published in "Is your secure test infrastructure secure enough?: Attacks based on delay test patterns using transient behavior analysis"[4] and then extended in "New Approaches of Side-Channel Attacks based on Chip Testing Methods"[1] and is joint work with co-authors Dennis R. E. Gnad, Jonas Krautter, and Mehdi B. Tahoori.

With increasing complexity of modern digital chips, the test costs also rise. In order to reduce these test costs, the use of DfT techniques has become de facto, improving the testability, diagnostics, test time and reducing the number of required test pins. But besides the benefits, a DfT infrastructure and particularly scan paths, which provide access to the internal states of the circuit, can also be used for security breaches. As a matter of fact, scan chains for secure devices open a backdoor for security threats, so-called "Scan attacks" [96]–[101]. Such attacks have already been exploited, for circumventing copyright protection in devices such as video game consoles [102], [103] or to establish fake base transceiver stations to attack mobile phones [104]. In addition, fabrication processes are getting more complex, too, and force most companies designing SoCs to no longer maintain their own fabrication unit (foundry or fab) or own testing facility. Therefore, to prevent leakage of any secret information while tests are performed, it is essential to secure the test infrastructure of the device.

The security risks associated with the on-chip DfT infrastructure, as well as untrusted testing facilities, can be prevented by a wide range of solutions to secure on-chip DfT and Test Access Points (TAPs) [98], [105]–[111]. Such techniques typically ensure that secret data and keys are not directly accessible through scan chains [105], [106], or they try to fully encrypt the test ports [108], [109]. Either way, it is guaranteed that the publicly accessible outputs or test interfaces do not contain any secrets and cannot leak secret information. Completely disabling the DfT infrastructure after manufacturing test is theoretically possible [112] but no longer an option due to functional safety requirements for in-field testing [113].

Usually, protected test infrastructure only secures any direct access to the secret information by test. So, the tests on publicly accessible (insensitive) outputs are not considered here in terms of security. For example, an output of a cryptographic algorithm which is already encrypted has not been seen as a potential information leak. However, the transient states of such insensitive outputs, which can be obtained for example via delay testing, can be used to reveal the secret information. If such transient information is accessible through the test infrastructure, it makes the test infrastructure insecure. In this

paper, we show that various types of delay testing which give insight into transient circuit behavior like Fmax testing [114]–[117] as well as transition delay, path delay, and Small Delay Defects (SDDs) tests [118], [119] can be utilized to execute attacks and consequently exfiltrate secret information from public (insensitive) outputs whose normal logic states are assumed to be secure. The presented attacks can be executed on an untrusted test floor or on-chip testing infrastructure, even when the DfT infrastructure implements encryption [120].

For our attacks, we utilize a profiling stage to characterize the *victim device* first, and then use the acquired information to recover its secrets later in a statistical template attack. We performed the proposed attacks on the hardware implementation of a symmetric encryption protocol by observing the transient behavior of the encrypted (presumably secure) outputs. The attacks are successful even when the template is built on one chip and the attack is performed on another chip, with slightly different transient behavior due to process variations. Also, the runtime variations, measurement noise, and inaccuracies do not impact the success of the attacks. Additionally, the attacks are even successful when the measurements of the transient output behavior are down-sampled by a factor of 10 to account for inaccuracies in delay testing measurements. Our results confirm the power of these attacks and highlight the fact that new approaches are needed to secure testing infrastructure.

The contributions of this chapter can be summarized as follows:

- Feasibility analysis of gathering transient circuit output behavior information in Test Environments, and with relation to Automatic Test Equipment (ATE) capabilities.
- Threat model based on an attacker who can observe transient circuit output behavior through test.
- Evaluation of FSA attacks on the measured traces.
- Introducing new attacks on transient circuit output behavior traces.
- Evaluation of presented attacks on traces with lower resolution.
- Introducing an attack on compacted test response signatures (i.e. MISR).
- Introducing an attack on pass/fail test responses.
- Evaluation of robustness of presented template attacks to runtime variations with traces gathered at different environmental temperatures.

In the remaining paper we first give the necessary background as well as explain our threat model in Section 3.1. In Section 3.2 we show how transient circuit output behavior can be used to reveal secrets from a Circuit Under Test (CUT), followed by a description of the exact setup we use in our experiments in Section 3.3. The results achieved with our attacks are presented in Section 3.4 and discussed in Section 3.5. Finally, the paper is concluded in Section 3.6.

3.1 Preliminaries

3.1.1 Attacks based on Test Access

The scan-chain access, if unrestricted, is an easily exploited attack vector to gather sensitive information [98], [107]. In the seminal work by Yang et al. [96], scan chains have been identified as an attack vector to directly extract secret information from a circuit, which subsequently led to various other attacks [101], even on real applications [102]–[104].

Therefore, securing test data or TAPs has become important, which early on has been done mostly through obfuscation [106], but later also cryptographic approaches [101], for instance, encrypting the test vectors [110] or the full scan chain [108], [109], [111]. There has also been one strategy proposed which only tests with a single fixed test key [105] at the disadvantage of reducing the possible test coverage. Some industrial approaches propose to completely disable the TAP after the full product has rolled out [121], which is not always a feasible solution. In the first place, this method can still not defend against the threat of an untrusted test floor [120], and it also does not comply with functional safety testing standards for in-field testing and diagnosis according to ISO 26262 [113].

3.1.2 Transient circuit output behavior

The combinational part of the sequential circuit is realized by a network of interconnected logic gates. These gates have one important property, which is their specific propagation delay. Due to this delay, after a change at an input of a gate, it takes a specific amount of time until the output of this gate changes. Therefore, the time in which the output at the endpoint of a combinational logic block switches to its desired value is dependent on the activated paths. This activated path is in turn dependent on input and internal data of the circuit, and can be used by an attacker to reveal the secret, like the FSA attack exploits.

Furthermore, it can happen that the input a of a gate always arrives earlier than the input b due to different path delays in the combinational logic of the circuit leading to a and b, correspondingly. Typical combinational circuits have many stages of logic blocks, with different internal delays. Therefore, the output at the end of a combinational logic block can switch more than once within a clock cycle. Such events are known as glitches and are possible due to inevitable or very hard to prevent hazards in most circuits. The occurring of these unnecessary transitions and their timings depends on the activated paths. By that, the transient circuit output within a clock cycle can also be used as a source of side-channel information and can be utilized for a template attack.

In general, once the physical implementation and thus all internal delays of a circuit are fixed, they define a specific, unique data-dependent behavior. This behavior can be exploited by an attacker using the actual chip or gate-level timing simulations.

As an illustrative example of transient circuit output behavior and its data dependency, we imagine a hypothetical circuit with multiple inputs and one output. We observe a

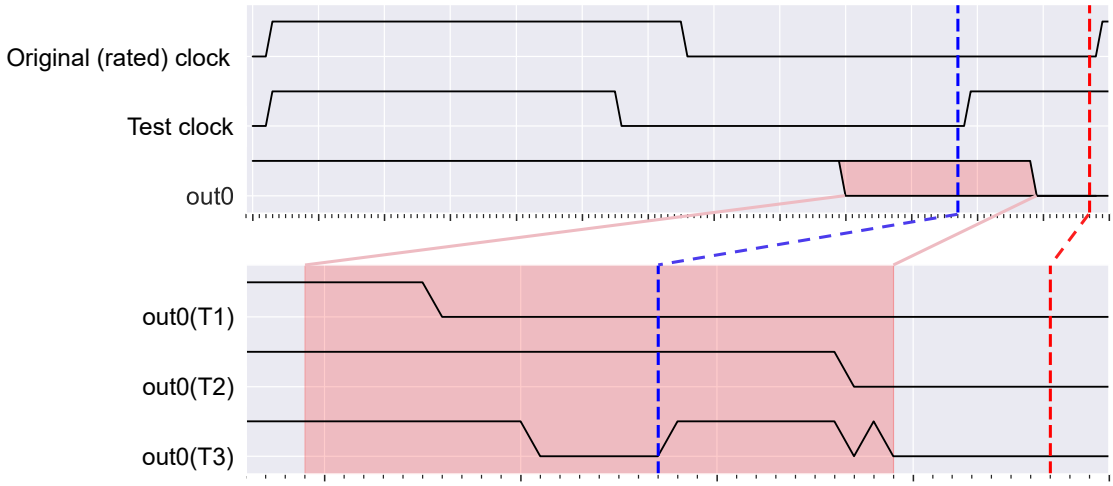


Figure 3.1.: Example of possible transient circuit output behavior occurring because of their dependency on the input of the circuit (hypothetical input transition 'T1','T2', or 'T3').

transition from logical 1 to logical 0 at the output of the circuit, as illustrated in the upper half of Figure 3.1. The red marked segment is an interval where output transitions can take place, dependent on different input transitions. The red dashed line is indicating the time point according to the Original (rated) clock and the blue line according to the test clock where the outputs need to settle down to the correct values. So, in the case of the Original (rated) clock being used, there are no faults, because the signal will be settled to the final value before the next clock edge. But if the Test clock is used, the clock cycle gets shorter and the output can get faulty, because the clock cycle end is within the period where transitions could happen. This illustrates how to measure the whole transient circuit output behavior by shifting the clock.

Such output transition can happen for several input transitions, and we name three of these hypothetical input transitions T1, T2 and T3. For the input transitions T1, T2 and T3 the output transition from 1 to 0 can be slightly different dependent on which paths were activated, as shown in the bottom half of Figure 3.1. If the activated path is short, as in the case of out0(T1), the output settles earlier on the final value, and if the activated path is longer, as in the case of out0(T2), it settles later on the final value. Additionally, if paths of different lengths are activated by a single input transition, multiple transitions can occur before settling to the final value, as shown in the case out0(T3). Each of these unnecessary transitions and their timing can reveal some additional information about the particular activated paths, which can be used by an attacker to recover the inputs of the circuit, as shown later in the paper.

3.1.3 Obtaining Transient Output Behavior using Fmax and Delay Testing

Many testing techniques obtain information about the transient behavior of the CUT including the timing (delay) of various paths in the CUT. This information is usually not

enough to cover the whole period of signal transition at the output, which can be even impossible to collect depending on the test technique and design limitations. However, an attacker can adjust the test flow to get more transient information from the circuit output or perform a successful attack with partial transient information at the cost of using more traces. For example, using ATE can allow more adjustments to the test flow by an attacker and testing using Built-In Self-Tests (BISTs) can be restricted to a few time stamps of the slack interval. The amount of revealed side-channel information is dependent on the measurement resolution and the length of the sampling. These define how many and how precise the signal transitions at the output can be captured.

One of the test techniques which is suitable to collect transient information is Fmax testing based on at-speed testing. At-speed testing refers to the test being performed at the system-rated clock, and in Fmax testing, the clock frequency of the circuit is gradually increased until it fails to meet the timing requirements. This is used to do the speed binning of the circuits [114]–[117]. The test results obtained from at-speed Fmax testing can be easily converted to the needed fault sensitivity for an FSA attack. The conversion is as simple as using the clock frequency at which a fault is detected, or calculating the time difference of the shorter clock cycle in relation to the original (rated) frequency. These can then be used as the input dependent fault sensitivity values of the critical path for the FSA attack. Conventionally, the test is only performed until the first fault occurs. However, an attacker could also continue increasing the frequency until a specific output gets faulty to determine its fault sensitivity. If necessary, the fault sensitivity of every single output can be gathered in this way. Furthermore, storing the output values every time after increasing the frequency can reveal the whole transient circuit output behavior or at least a part of it, in case the circuit completely fails at some point due to design limitations.

Delay testing methods and especially SDD testing can also provide detailed information about the transient behavior of the CUT [118], [119]. In general, delay testing is used to verify that a circuit meets its timing specifications. Depending on which paths need to be activated, two patterns are applied to the CUT, one after another: The *initialization vector* to initialize the CUT and the *test vector* to launch the transition. However, it can be helpful to use faster-than-at-speed testing which uses a faster rated clock frequency to also identify SDDs which can stay undetected during at-speed tests with the originally rated clock frequency. Therefore, in SDD testing, potential future chip failures due to aging effects can be identified by measuring the margin for timing failures for particular combinational paths. To measure the margin, paths from input to output are activated either by scanning through the slack interval [118] or activating particular hazards [119]. Therefore, the entire transient behavior of the circuit can be obtained using such small delay testing methods, independently of actual defects. This information reveals how the values of the output pins are changing during the clock period, and can be used by an attacker for a template attack, as shown later in Section 3.2.

Such precise timing information can be obtained with state-of-the-art ATE which provides precise timing measurements and can be in the precision range of a few picoseconds [122], [123]. Alternatively, it can be done using on-chip test infrastructure using BIST [124], or programmable on-chip capture [125] and measurement [126] schemes. We show

in Section 3.4.5 that even BIST data compacted in a standard Multiple Input Signature Register (MISR) scheme can reveal side-channel information when any change in its output can be observed.

Such types of measurements are also feasible in FPGAs, where a delay line can be employed [127].

3.1.4 Threat Model Assumptions

There are many approaches which have been proposed to secure test access [98], [101], where most of the solutions restrict access to the TAP in various ways. However, not all of these solutions might be completely practical in all situations. In some scenarios, functional safety standards such as ISO 26262 can mandate the access to delay testing information in order to predict chip health or allow fault containment [113]. This necessary delay test data requires information beyond, for instance, a simple PASS or FAIL output from a BIST module. An output with such reduced information had also been considered a security feature of earlier BIST concepts [101], as involuntary exposure of internal states is prevented. Actual industrial BIST implementations typically use a test compactor (i.e. MISR scheme) that can still reveal some information [128], [129], and there are no widespread standard security features for securing TAP available yet [129]. Because of ISO 26262, a fully restricted test infrastructure that still fulfills all safety requirements cannot be expected, as there is typically a trade-off between security and testability [113].

Based on this, our two main assumptions to make the proposed attacks possible are:

- The attacker has access to a copy of the device which is under their full control (*templating device*), including the possibility of setting any key (or alternatively, an attacker has access to the netlist of the circuit and uses simulation).
- The attacker has access to the outputs and according transient circuit output behavior of the *victim device*.

In Figure 3.2 we visualize our threat model, showing various possible attack vectors to the CUT. A copy of the *victim device* is necessary for template generation using a black-box model (i.e., no access to gate-level netlist) [87]. If a copy of the device is not available, having access to the gate-level netlist or being able to reverse-engineer it from layout/mask could also be sufficient [130].

The transient circuit output behavior can be obtained, via ATE on a test floor of an untrusted foundry or via on-chip measurements, as discussed later in Section 3.1.3. There is no need for any other additional information about the internal states of the circuit like JTAG or scan chain outputs.

We also consider that the secret value does not change during an attack, but there are no restrictions on how or where the secret is stored. The only exception applies when the secret is stored in read-only memory, since the attacker needs the ability to change the

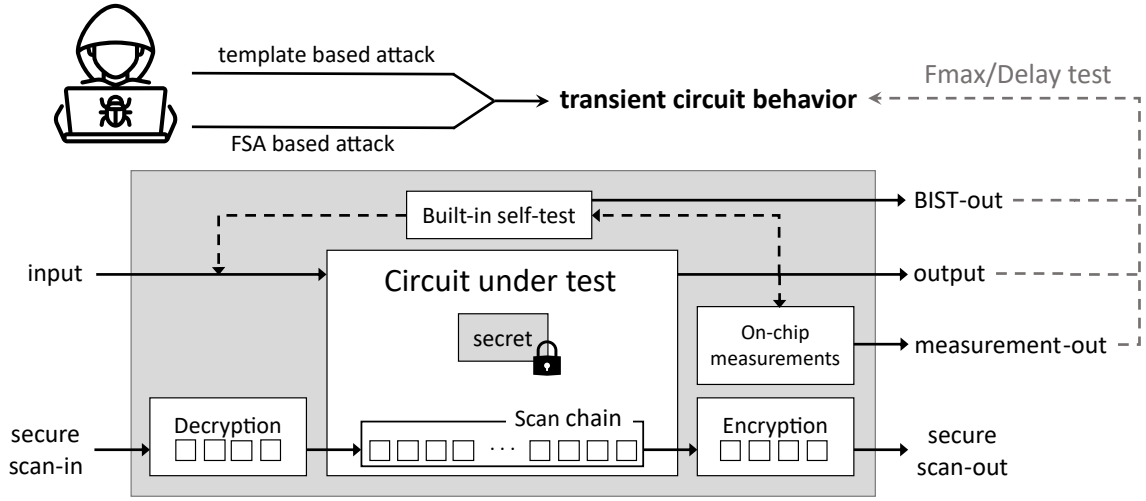


Figure 3.2.: Threat model showing the Circuit Under Test (CUT) and possible attacks based on delay measurements through various channels such as BIST, direct output, other on-chip measurements. The attacker uses profiling/templating information from another device.

key for the template generation process. For this case, simulations can be used for the template generation, which makes knowledge of the circuit netlist mandatory.

3.2 Methodology

3.2.1 Attack Flow

Like in other side-channel analysis attacks, we use some additional information to correctly guess parts of a larger key. Our kind of used side-channel information is the transient circuit output behavior. The transient behavior of a circuit output pin is mainly data dependent on the circuit input pins, which are connected to this output pin through a path in the circuit. If circuit inputs are not connected through a path in the circuit to an output pin, these can be seen as not affecting its transient circuit output behavior. Therefore, the attacker can utilize the divide and conquer paradigm to attack (or create a template for) a part of the circuit at a time. As an example, we examine the last round of a round-based AES hardware implementation. The boolean XOR operation, which is used for the AddRoundKey operation, adds a connection for each output bit to the input bit of the round key and a corresponding output bit of the ShiftRows operation. ShiftRows does not add any new connections, and the SubBytes operation adds a connection from each AES SBox output bit to its eight input bits. Consequently, an adversary can attack (or create templates for) each output byte separately, because it is also connected to only one byte from the round key and one byte from the round input.

The attacks we use rely on template generation and consist of two stages: Profiling phase (Algorithm 1) and Attack phase (Algorithm 2). For that, we assume that an attacker can use a copy of a *victim device* under their full control, including the ability to set

the secret key for the profiling phase. The template is needed for precise prediction of data-dependent transient circuit output behavior in the attacking phase, but does only need to be generated for the observed output transitions of the *victim device*. An output transition is a pair of previous and current output values between which we gather the transient information. A template entry is then a pair of consecutive outputs (c_0, c_1), used key k and a trace s describing the transient output behavior. This process is similar to building Fault Dictionary (FD) for fault diagnosis [131].

The input transitions for template generation can be either predefined by the specific test flow (Automatic Test Pattern Generation; ATPG) or random (BIST generated patterns). In the case of using random patterns, the profiling phase can be done even after all traces to attack are already collected. The input transitions for the templating device are then calculated from the obtained output transitions in combination with all possible subkeys. When attacking an AES cryptocore, for instance, only one key-byte is attacked at a time, which means there are only 256 possible subkeys (the remaining key bytes can be padded with zeros). Then, transient circuit output behavior can be captured for all these input transition combinations and stored as a template. To make the template more robust against runtime variations, it can be captured multiple times and averaged. Thus, the template will be a trace of real-valued numbers between 0.0 and 1.0, and not just boolean. In the attacking phase, we finally calculate the Pearson correlation coefficient ρ of the traces from the victim device with according template traces for every possible guessed key. The correct key is considered to have the highest positive ρ value.

The described attack scheme can be compared to the process of building a Fault Dictionary (FD) for fault diagnosis [131]. In the latter, the FD is built through extensive simulation of the CUT, whereas we build the template directly through measurements on a second device. For fault diagnosis based on cause-effect, the FD is used to identify the fault that caused an observed effect in the outputs. Here, we instead make use of the data dependency of the transient output behavior to extract secret information.

Algorithm 1 Template building

Input: Set of ciphertexts pairs
$C = ((c_0, c_1)_0, (c_0, c_1)_1, .., (c_0, c_1)_n)$
Output: $Template$

```
for all (c_0, c_1) ∈ C do
    for all k ∈ {0, .., K_max} do
        (m_0, m_1) ← decrypt((c_0, c_1), k)
        s ← RecordTrace((m_0, m_1), k)
        Template ← Template ∪ ((c_0, c_1), k, s)
    end for
end for
return Template
```

The main difference between the various proposed attacks is the type and the amount of required transient circuit output behavior information and how they are processed.

Algorithm 2 Template matching

Input: Ciphertext pairs C, according traces S
Output: Key k

```
for all k ∈ {0, .., K_max} do
    ρ_k ← PearsonCorrelation(S, Template(C, k))
end for
k ← ArgMax(ρ)
return k
```

3.2.2 Template attack on FSA

While the original FSA attack is based on active fault injections on the *victim device*, we use a passive side-channel approach based on analysis of transient behavior of the device using the data available through tests. The transient circuit output behavior traces from both the *templating device* and the *victim device* are reprocessed in such a way that the fault sensitivity is stored for each output pin. In that case, fault sensitivity is the time difference between the end of the clock cycle where all outputs are stable and a time point within this cycle where an output pin becomes faulty. Therefore, the output pin with the smallest time difference to its fault refers to the critical path w.r.t. the input transition. An attacker can use this fault sensitivity of the critical path according to the inputs transition (path delay test on most critical output) or fault sensitivity of any specific bit output (path delay test on a specific output pin) or all outputs together (path delay test on all output pins).

3.2.3 Template attack on full transient circuit output behavior traces

The full transient circuit output behavior traces store the entire information of all signal transitions and their timing from the previous to the actual clock cycle for every single output pin. Such a trace is represented as n lists of single bits, one list for each circuit output pin. Each list represents transient states of the circuit at different points in time between the clock cycles. One such list can be seen as a single sub-trace, where all sub-traces together build a whole trace. For better handling, we concatenate all sub-traces belonging to a trace into one large list, which is illustrated by the following equation where $b_{i,j}$ is a single data bit according to the i-th subtrace and j-th timestamp.

$$\left.\begin{matrix} b_{0,0}, \cdots, & b_{0,m} \\ \cdots & \\ b_{n,0}, \cdots, & b_{n,m} \end{matrix}\right\} = b_{0,0}, \cdots, b_{0,m}, \cdots, b_{n,0}, \cdots, b_{n,m}$$

This way it can be easily correlated with another large list belonging to another trace. Correlating more than one trace at a time can then be handled as concatenating these large lists again, or averaging results of single correlations. The correct key here is also considered to have the highest positive value of ρ.

Although capturing all transitions within the transient circuit output behavior is possible with the resolution of state-of-the-art ATEs [122], [123], we show that even down-sampling the transient behavior with a significantly reduced resolution is still sufficient for the success of this template attack.

3.2.4 Template attack on specific time point of transient behavior

An attacker can also perform transition fault tests, in which the circuit output is evaluated at a single time point after the test pattern pair is applied. It is important that enough side-channel information is available at a chosen time point for the attack to succeed. The templating phase can be used for finding a suitable time point. Based on these time point samples, we can then perform the attack similarly to the one presented in Section 3.2.3, where the values of output pins at a single time point instead of a whole trace get correlated.

3.2.5 Template attack on compacted test response signatures

For acceleration and reducing the test overhead, the test response can be compressed in both the space and time dimensions to a compact signature [132]–[134]. Such a signature can be generated, for example, by a MISR scheme and does not contain direct information about the test results. However, we show that our template attack is still possible using this kind of reduced side-channel information. To evaluate this approach, we reduce the collected data to only eight points during the sampling period from all circuit outputs and compress it to a single 8-bit signature with a MISR scheme. It compresses 64 bits of transient circuit output behavior information to an 8-bit signature, without being reversible. The attack flow stays the same as in Section 3.2.3 except that the full traces are replaced with signatures.

3.2.6 Template attack on pass/fail test responses

Some test responses can be limited to the information if the test was passed or failed, without giving any information about actual output states. The test response would be passed if the outputs were correct and failed if the outputs were wrong. For our attack, we assume that test responses are available for path delay tests on several different frequencies (clock cycle durations) and reveal if the outputs were correct at different time points of transient circuit output behavior. This information can then be handled as side-channel information to reveal the secret, although the output values stay unknown. Like in previous attacks, the attacker first needs to build a template for all used test patterns in combination with all possible subkeys. In the following step, the observed pass/fail test responses are statistically matched with the template. Here, the correct key is also considered to have the highest positive value of ρ.

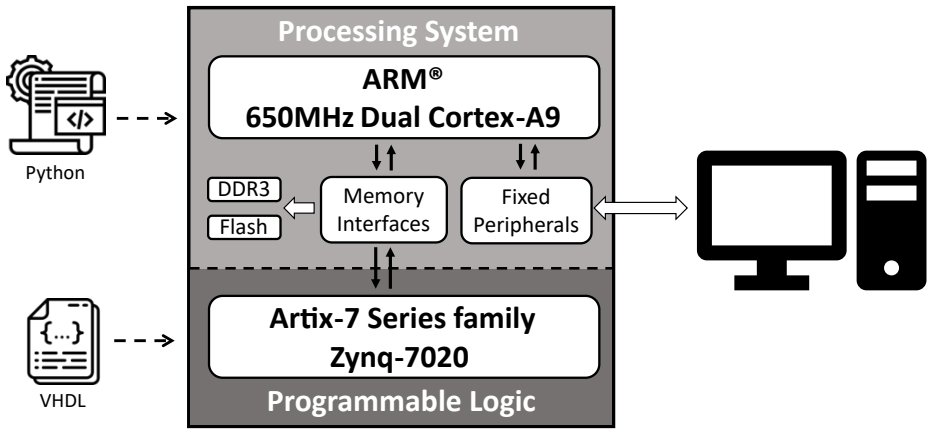


Figure 3.3.: Simplified overview of the Experimental Setup using the Xilinx PYNQ-Z1 board.

3.3 Experimental setup

To analyze the usability of this attack on real hardware and at the same time have full control over the transient behavior analysis, we chose an FPGA setup. The device used in our setup is the Pynq-Z1 board, which incorporates a Xilinx Zynq Z-7020 ARM/FPGA APSoC, a dual-core Cortex-A9 processor and Artix-7 family programmable FPGA logic. We use two copies of these devices – the first one is considered the *victim device* and the second one is the *templating device*. The attacker has full control only on the *templating device*, but can observe transient circuit output behavior of the *victim device*.

The FPGA setup is implemented as a *Pynq overlay* using Xilinx Vivado, and can be controlled through Python scripting. Two CPUs are running a Linux operating system, and a web server hosts an environment to run Jupyter Notebooks for the used Python scripts. These CPUs are connected with the Artix-7 FPGA fabric, and through that the FPGA can be controlled from the network.

For additional experiments at 0 °C, 45 °C, and 70 °C, we used LabEvent T/210/40/EMC/M climate chamber from Weiss Technik, which allows temperature tests in the range from -35 °C to +130 °C.

3.3.1 Acquisition of transient circuit output behavior with an observable delay line

With FPGAs, we have the opportunity to use their flexibility to build on-chip sensors for delay measurements. Such delay measurements on the clock signal were used in recent works [135], [136] for measuring the power consumption of the circuit. This can be used by an adversary for side-channel analysis [136] or as part of a hardware security primitive detecting malicious voltage fluctuations [137], [138]. However, we use a delay line sensor on an output signal of the CUT to measure its transient behavior. This delay

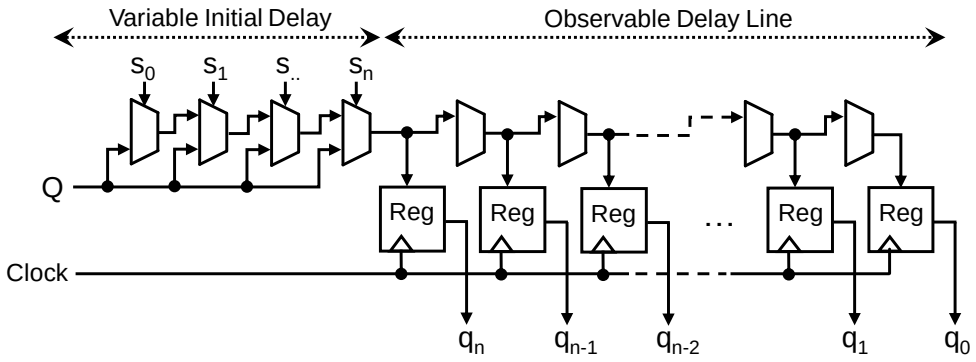


Figure 3.4.: Delay line used in this work with an adjustable initial delay. The signal Q at the input is one of the combinational outputs from the measured circuit, and thus transient circuit output behavior can be observed by looking at the outputs q_0 to q_n, with q_0 being the *oldest* value in time.

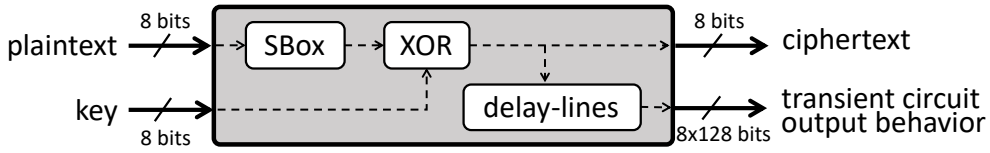


Figure 3.5.: Overview of method to measure transient circuit output behavior in 8-bit AES SBox + XOR circuit.

line is built using inherent transistor delays, replicating what can be achieved with on-chip test equipment [127]. Similarly to the recent works, our delay line implementation consists of two main parts, with the basic principle shown in Figure 3.4: Variable initial delay and observable delay line. Both can be built out of carry-chain primitives in Xilinx FPGAs, which are essentially chains of MUXes. Depending on the respective MUX input selection through the signals s_0 to s_n, the measured signal Q enters the variable initial delay at a later or earlier time. Q at different points in time is then finally captured in the registers of the observable delay line q_0 to q_n. These registers need to use the same *Clock* as the final registers of the circuit feeding the Q input, i.e. the output registers of a cryptographic circuit.

For calibration, the initial delay has to be adjusted such that the output of q_0 shows the correct output of the previous clock cycle, while q_n shows the correct output of the current clock cycle that is also captured in the output registers. Thus, in q_1 to q_{n-1}, the transient output behavior within one clock cycle is captured.

We assume a measurement resolution of our delay line on the used platform of about 11.5 ps [135], where the timing analysis estimate is 13.25 ps. As explained in Section 3.1.3 similar traces can be also obtained through recent ATE or delay testing BIST which can offer similar or even better measurement resolution [122], [123]. Therefore, it is realistic to obtain traces in a similar time resolution as the proposed traces. Additionally, the attack is still successful if we down-sample our traces with a factor of up to 10x, as shown later. So, we can assume that ATE or delay testing BIST measurements with resolutions up to 100 ps should also be feasible.

3.3.2 Circuit under test design on the FPGA

For analysis, we developed two basic logic implementations of SBox + XOR combination. The first one is a 4-bit SBox + XOR combination of the PRESENT cipher, because it makes it feasible to exhaustively check all possible input transition combinations for the acquisition of corresponding transient circuit output behavior. The second and main implementation is an 8-bit SBox + XOR combination of the widely used AES cipher. We provide an overview of the basic logic in Figure 3.5, which takes an 8-bit plaintext to the AES SBox and the output is then provided to an XOR together with the key, which results in an 8-bit ciphertext.

Because of the delays in the circuit, transitions occur at some points within the clock cycles, depending on the input transitions. Therefore, to make the observation of these transient circuit output behaviors possible, each bit of SBox output is also connected to an instance of the delay sensor with 128 bits output each. The 4-bit implementation is analogous to the 8-bit with one difference: Delay sensor lines with a length of 64 were found sufficient here, such that all transient values could be captured.

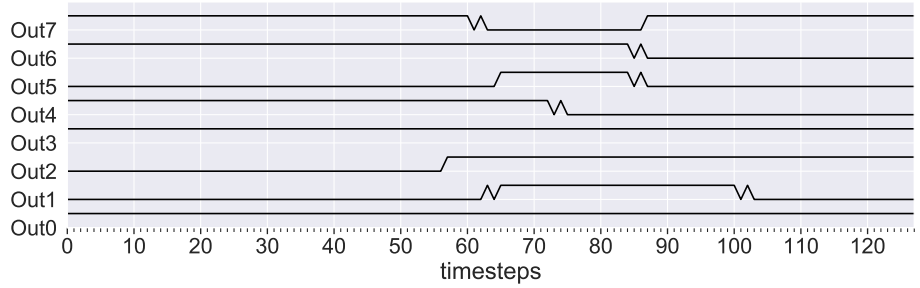
Next to the SBox + XOR and delay sensor lines, a state machine is implemented that performs tasks such as listening to the commands from Python scripts on ARM, setting inputs to SBox + XOR at the desired clock cycle, temporarily holding output values, and communicating them to the processing system.

3.4 Results

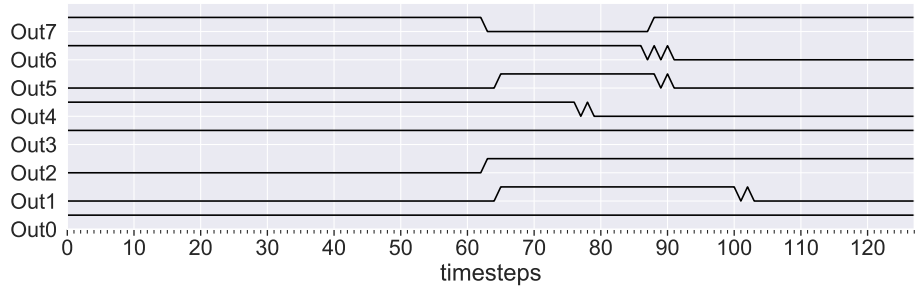
In this section, we present the results, which are all based on measurements on the actual Zynq Z-7020 FPGA devices detailed in the Experimental Setup. Most of these results are summarized in Table 3.1. We executed all described attacks on both of our setups, 4-bit PRESENT SBox + XOR and 8-bit AES SBox + XOR. The results on 4-bit PRESENT are included only in Table 3.1, otherwise we focus on the 8-bit AES SBox + XOR setup in the following subsections and present its results. For all the detailed results presented in this section, we always used two different FPGA Boards. One of them was used to collect several measurements for template generation (i.e. the *templating device*), and from the other, few traces were collected to attack it (i.e. the *victim device*). The respective amount of traces needed for successful attacks on the victim device are summarized in Table 3.1. These are the maximal number of traces needed for successful correlation with the correct key in a larger set of our experiments.

3.4.1 Transient circuit output behavior traces

Figure 3.6(a) illustrates an example of a properly recorded transient circuit output behavior trace on an FPGA with 8-bit AES-SBox + XOR setup with eight 128-bit long delay sensor lines, with each of the delay sensor lines corresponding to an output pin of the 8-bit



(a) A trace captured on *templating device*.

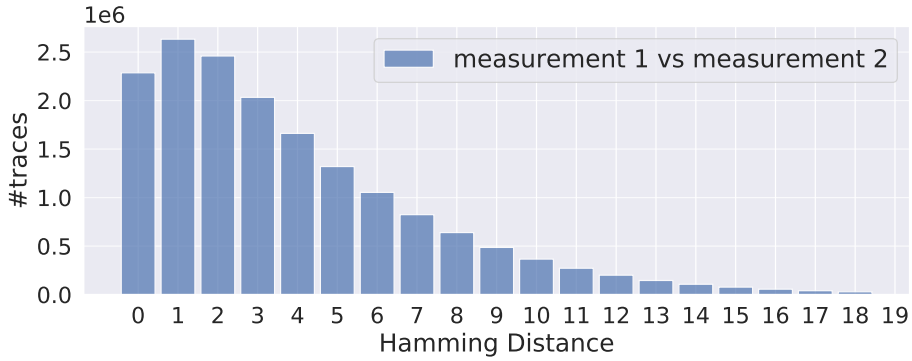


(b) A trace captured on *victim device*.

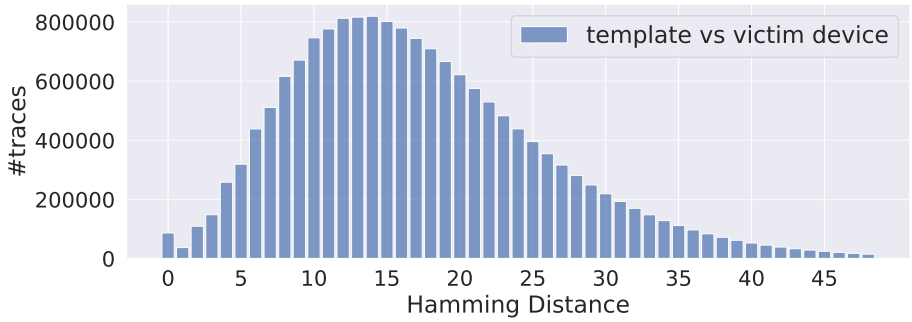
Figure 3.6.: Example transient circuit output behavior traces captured on real FPGAs (*templating device* and *victim device*) for key = 67 and input transition from 24 to 196.

circuit. Recording such traces several times with the same input transitions and on the same device does not always result in 100% identical traces. This is caused by runtime variations like voltage fluctuations, coupling effects of adjacent wires, and other noise sources such as clock jitter and temperature variation. Therefore, for building the templates, we record 100 traces for every needed input transition, and average them. The histogram in Figure 3.7(a) shows the distribution of Hamming Distances (due to runtime variations) between repeated measurements with the same input transitions on the same device for all possible combinations.

Next to runtime variations, chip-to-chip process variations also influence circuit output behavior and measurements of it, leading to differences in captured traces. Figure 3.6(b) illustrates another example trace with the same input transition as Figure 3.6(a), but recorded on another device. As can be seen in this example, there are differences, but the traces are still very similar in general. The second histogram (Figure 3.7(b)) shows the distribution of Hamming Distances between repeated measurements with the same input transition, but on two different devices. Additionally, to runtime variations, this histogram also shows the impact of chip-to-chip process variations. In Figure 3.8, we additionally illustrate chip-to-chip variations depending on different test patterns, i.e. all 256 different key bytes as input to our 8-bit setup, for both of the FPGA devices. The y-Axis shows the respective *fault sensitivity* of the activated path, depending on a test pattern leading to the same output ciphertext, for all different key inputs.



(a) Influence of runtime variations on one FPGA (intra-device).



(b) Influence of manufacturing process variations and runtime variations on two different FPGAs (inter-device).

Figure 3.7.: Histogram for influence of manufacturing process variations and runtime variations to traces with Hamming Distance as metric. Obtained through real measurements on FPGAs. (based on 2^{24} traces; 1 trace = 1024 bit = 8 x 128-bit sub-traces)

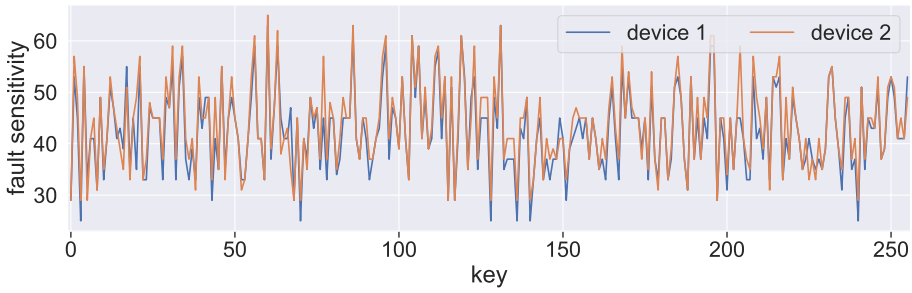
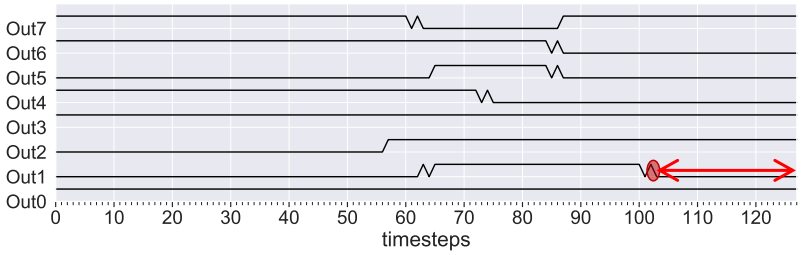


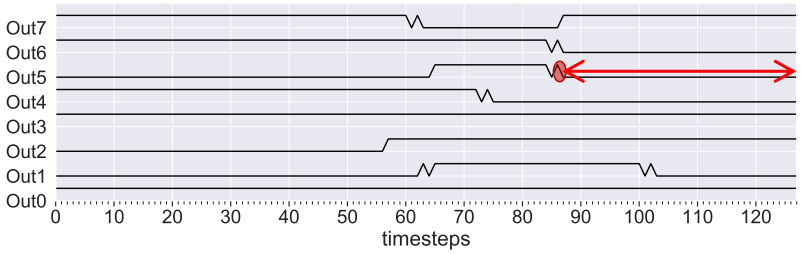
Figure 3.8.: Fault sensitivity values according to the most critical path delay of the same ciphertext transition (from 57 to 114) for different keys acquired from real measurements on two different FPGAs.

3.4.2 FSA attacks

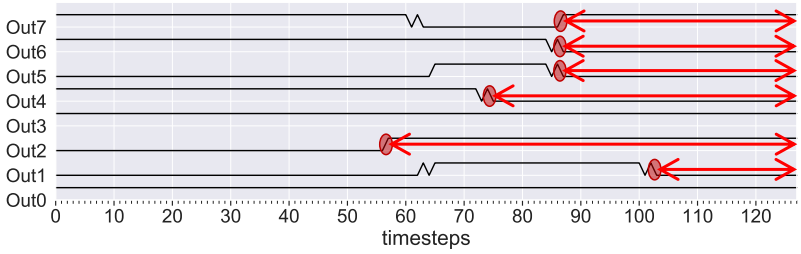
For the FSA attack, we first preprocess the transient circuit output behavior traces to get the timing information showing the respective fault sensitivities. We calculate this value for every single circuit output separately. For that, we look at the transitions inside the recorded trace, starting with the most recent value (the right side in Figure 3.9(a)-3.9(c)) in the direction of the oldest value (left side in Figure 3.9(a)-3.9(c))) until the first transition



(a) Fault sensitivity of critical output.



(b) Fault sensitivity of pin Out5.



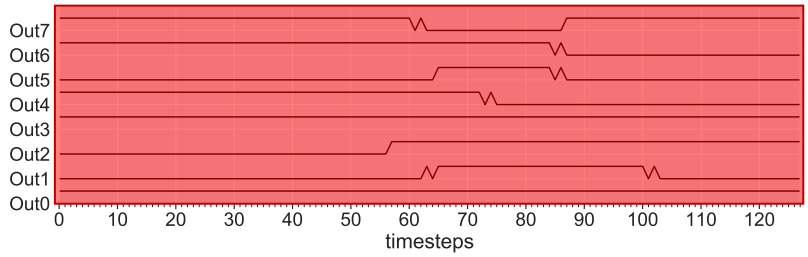
(c) Fault sensitivity of all output pins.

Figure 3.9.: Type of transient circuit output behavior information used for various proposed attacks. The basic trace in background is a real trace measured with our delay line. The only information needed for the attacks is marked red: Fault Sensitivity as timing information marked with red arrows.

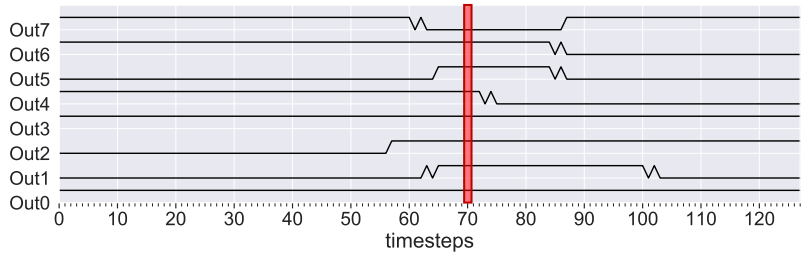
(from $1 \rightarrow 0$ or $0 \rightarrow 1$) occurs. With these preprocessed traces, we can execute three variations of the FSA attack, as we introduced in Section 3.2.2.

3.4.2.1 FSA attack on path delay test on most critical output

Our results show that with this approach the correct key can be recovered starting with 10 different traces. However, in our tests, the maximum number of traces which were needed to be successful in this attack was 22. Figure 3.11(a) illustrates an example of such correlation, the correct key has the highest correlation value after around 7 traces, and the difference to the wrong keys increases afterwards with an increasing number of transitions, making it even more distinguishable. With 15 transitions, it is clearly distinguishable from the incorrect ones.



(a) Full transient circuit output behavior trace.



(b) Specific time point of transient behavior.

Figure 3.10.: Type of transient circuit output behavior information used for various proposed attacks. The basic trace in background is a real trace measured with our delay line. The only information needed for the attacks is marked red: transient values at output pins marked as red rectangles.

3.4.2.2 FSA attack on path delay test on specific output

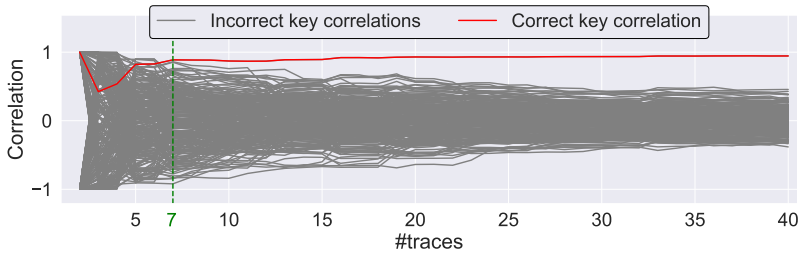
This attack needs, in general, a few more traces to correctly guess the key than in Section 3.4.2.1, and the difference in correlation coefficients between correct and wrong keys can also be significantly lower. However, in the range of 39-65 traces, dependent on the used output, we could always retrieve the correct key. An example of this attack using always the first sub-trace resembling the first output pin is shown in Figure 3.11(b).

3.4.2.3 FSA attack on path delay test on all outputs

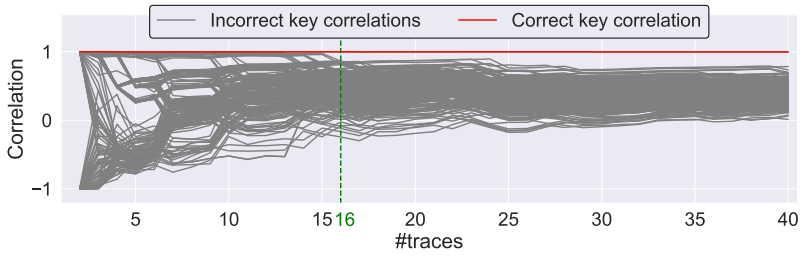
This is done by taking the average value of all correlation coefficients retrieved through the FSA attack on single sub-traces. This approach has turned out to be the most efficient one of all three. In our tests, this attack only needs 5 traces to recover the correct key and achieves more significant differences to the incorrect keys. Figure 3.11(c) illustrates such averaged correlation coefficients depending on the number of used traces.

3.4.3 Template attack on full transient circuit output behavior traces

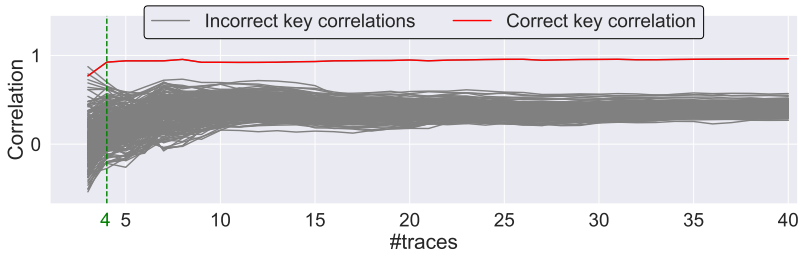
The second general type of attack we used is a template attack on full transient circuit output behavior traces (see Figure 3.10(a)). Since here the traces were not preprocessed and carry more information, it needs fewer traces to recover the correct key. For about



(a) FSA attack on most critical output.



(b) FSA attack on first output pin.



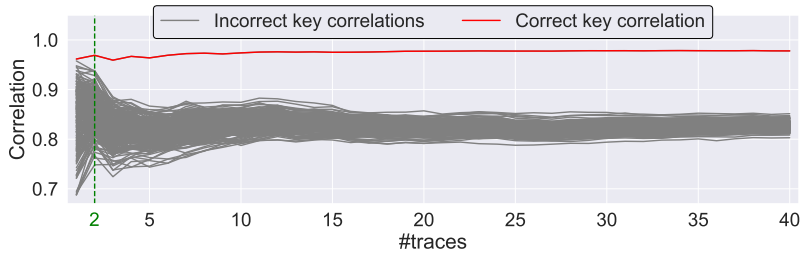
(c) FSA attack on all output pins.

Figure 3.11.: Results of FSA attacks based on path delay testing. Correlation values of the correct key (red) and incorrect keys (grey) depending on the number of used traces. The dashed vertical line indicates the number of traces from which the correct key has higher correlation coefficient than the others.

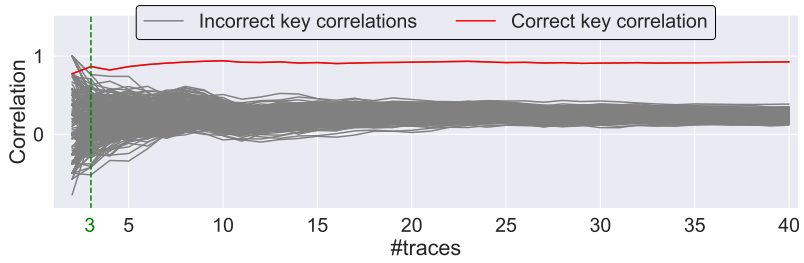
90% of our tests, a single trace was sufficient to recover the correct secret key. Adding just one more trace resulted in recovering the secret with only 2 traces in all tested cases. An example result of such an attack for different amounts of used traces is shown in Figure 3.12(a). This attack is also successful if transient information is restricted to a single output pin, which is one single subtrace in our case. Dependent on the output, in such a case there are 10 to 17 traces needed to reveal the secret.

3.4.4 Template attack on specific time point of transient behavior

When considering only a specific point in time in the transient output behavior (see Figure 3.10(b)) for analysis, the amount of traces for a successful template attack varies greatly. In Figure 3.12(b) we show an example of such an attack. Here, the correlation value at the specific time point 70 is shown over the number of used transitions for the



(a) Template attack on full transient circuit output behavior trace.



(b) Template attack on specific time point of transient behavior.

Figure 3.12.: Results of template attacks based on path delay testing. Correlation values of the correct key (red) and incorrect keys (grey) depending on the number of used traces. The dashed vertical line indicates the number of traces from which the correct key has higher correlation coefficient than the others.

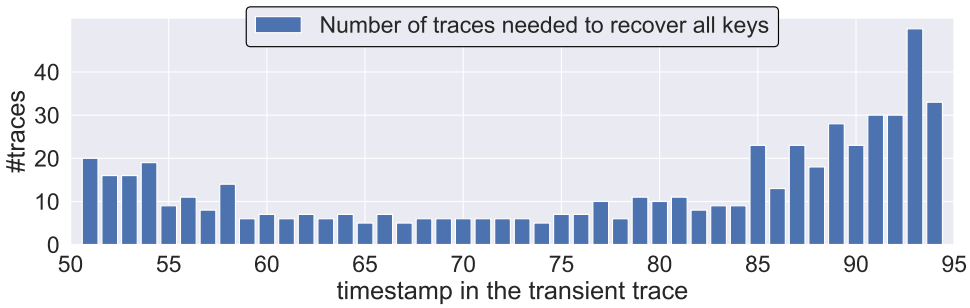


Figure 3.13.: Comparison of estimated amount of traces needed to recover a key for template attack on specific time point of transient circuit output behavior according to used time point.

attack. In this specific case, the maximum correlation for the used key is already achieved when using three traces. Over all the tested time points, we needed from 5 to 50 traces to recover all keys, as we show in Figure 3.13.

3.4.5 Template attack on compacted test response signatures

A compacted test response signature contains reduced side-channel information compared to a full transient behavior trace. For that, we analyze a template attack on a compacted test response signature following the MISR scheme (see Figure 3.14). In that case, still just a few traces are required for a successful attack. Figure 3.15(a) illustrates an example of such an attack, where the correct key has the highest correlation value already after

Table 3.1.: Attack results summary. Estimated amount of random traces needed for a successful attack.

		FSA attack on path delay test on: (cf. Section 3.4.2)			**Template attack on transient circuit output behavior:**	
		most critical output (cf. Section 3.4.2.1)	**specific output** (cf. Section 3.4.2.2)	**all outputs** (cf. Section 3.4.2.3)	**full trace** (cf. Section 3.4.3)	**specific time point** (cf. Section 3.4.4)
templating and *victim device* are the **same device**	4-bit setup	5 traces	7-12 traces	4 traces	**1 trace**	4 traces
	8-bit setup	10 traces	20-40 traces	4 traces	**1 trace**	3 traces
templating and *victim device* are **different devices**	4-bit setup	7 trace	15-20 traces	5 traces	**1 trace**	4 traces
	8-bit setup	22 traces	39-65 traces	12 traces	**2 traces**	6 traces

5 traces, from where the difference in correlation to the wrong keys increases with an increasing number of traces, making it even more distinguishable. However, it does not match with the template as well as previous attacks, since the correlation value of the correct key is significantly lower than 1. We estimate that this attack is always successful with about 18 traces.

3.4.6 Template attack on pass/fail test responses

For this attack we define our trace as a set of pass/fail path delay test responses (1 or 0) for different timepoints of transient circuit output behavior according to a single input pattern. The results here depend on both the number of such timepoints as well as the number of traces. Using a lower number of timepoints of transient circuit output behavior requires a larger number of traces for a successful attack, and vice versa. In our example attack we use traces with pass/fail data from 8 timepoints. In this case we estimate that the attack is always successful with 22 traces. Figure 3.15(b) illustrates an example of this attack, where the correct key has the highest correlation value after 10 traces.

3.4.7 Template attacks on traces with lower resolution

In addition, we also performed some tests of presented attacks on down-sampled traces. Since our traces are captured with a resolution of about 11.5 ps, we can show that these attacks are still efficient with a resolution down to 1/10 of the original. For this test, we simply took every n-th bit of our original trace, e.g. every 5-th bit to achieve a resolution with a factor of 1/5 to the original. Then we performed an FSA attack on critical path timing and a template attack on full transient circuit output behavior traces exactly the same way as before, but on these down-sampled traces. Figure 3.16(a) and Figure 3.16(b) show the corresponding results respectively. In general, the difference to the original trace (resolution 1/1) is not very significant and turns into needing just a few additional

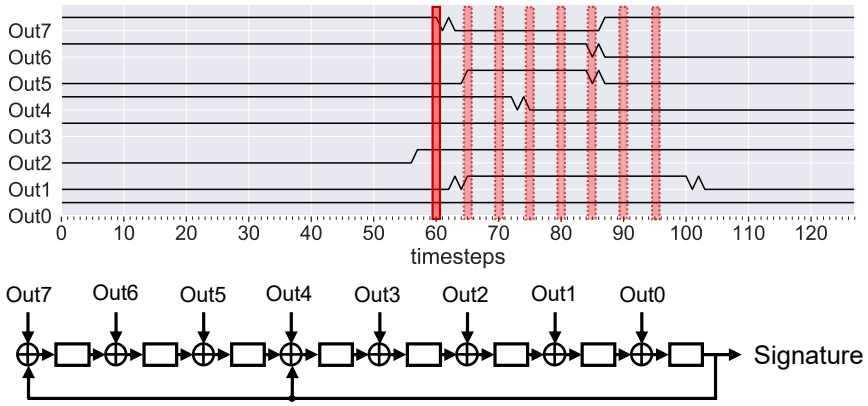
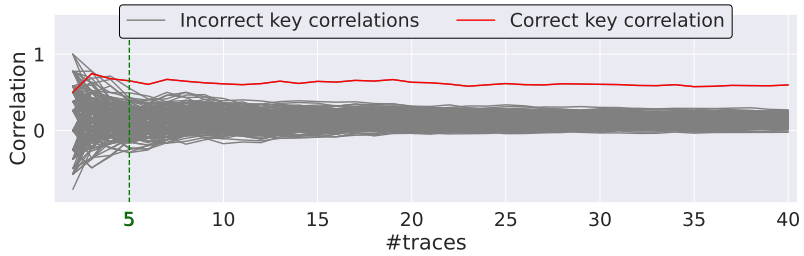
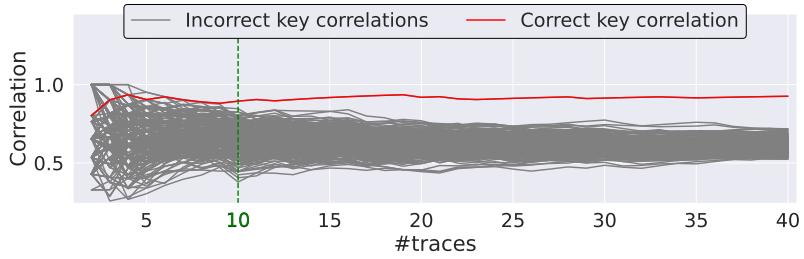


Figure 3.14.: MISR scheme for generating an 8-bit test response signature by applying transient circuit output behavior data from 8 timestamps, one after another.



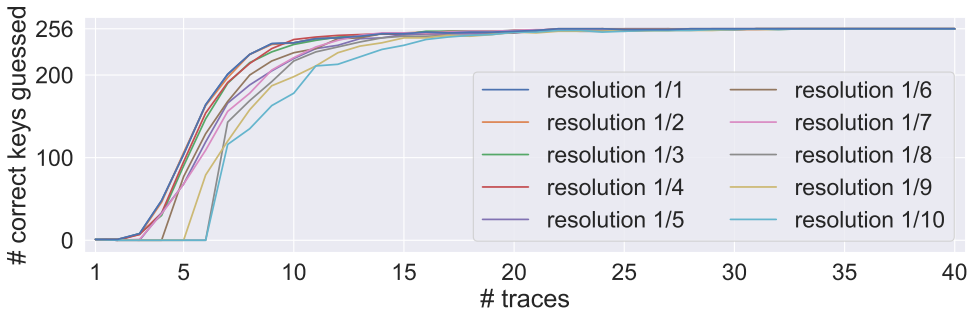
(a) Template attack on compacted test response (MISR).



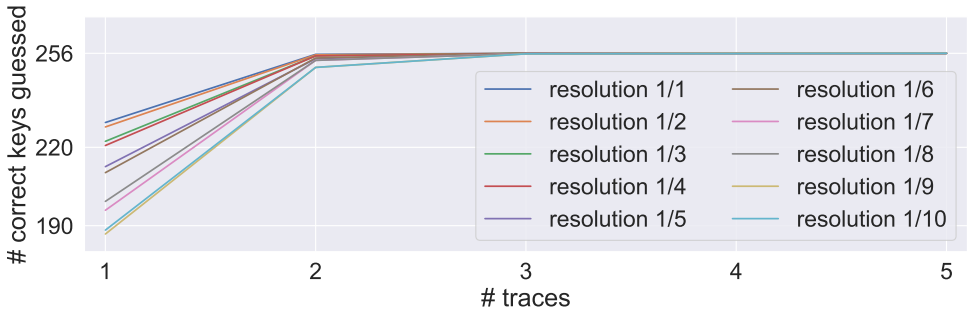
(b) Template attack on pass/fail test response.

Figure 3.15.: Results of various attacks based on path delay testing. Correlation values of the correct key (red) and incorrect keys (grey) depending on the number of used traces. The dashed vertical line indicates the number of traces from which the correct key has higher correlation coefficient than the others.

traces for a successful attack. For that, about 5 to 10 additional traces are needed for FSA on critical path timings, and about 1 to 2 for template attacks on full transient circuit output behavior traces. This shows the power of this attack even with lower resolution and noisier measurements, either on-chip or with an ATE.



(a) FSA attack on path delay test on most critical output.



(b) Attack on full transient circuit output behavior traces.

Figure 3.16.: Comparison of template attacks on down-sampled traces. The number of successfully recovered keys depending on the number of used traces according to the different resolution factors of the original trace.

3.4.8 Attacks on traces gathered at different temperature

The quality of measured traces is often dependent on the environmental temperature. To show the robustness of our attacks to runtime variations, we have repeated presented attacks where traces from *templating device* and traces from *victim device* were gathered at different temperatures. The attack flow always remained unchanged, including templates, which are based on the traces gathered at office temperature (about 22 °C). The traces from *victim device*, however, are newly collected at different temperatures in a climate chamber. We collected three sets of such traces, the first at 0 °C, the second at 45 °C, and the third at 70 °C. Figure 3.17 illustrates various examples of these attacks on victim traces gathered at 70 °C. In general, our results show that there are no significant differences compared to attacks where both templating and victim data were collected at office temperature. It is comparable to runtime variations of different data sets collected at the same temperature. Estimated amounts of such random traces gathered at different environmental temperatures needed for a successful attack for all presented attacks are summarized in Table 3.2.

3.4.9 Comparison of methods

If we compare the proposed attack methods to each other, all of them succeed with a really low amount of traces, which is significantly lower than the number of test patterns usually used for testing. However, there are also some differences like the type of needed transient circuit output behavior. This information is tightly coupled with the amount of repeated measurements (dependent on measurement type) and the template size. For this comparison, we assume that the attacker is doing Fmax-like measurements by increasing the frequency with a certain step to scan through the transient circuit output behavior. This step defines how fine-grained the transient circuit output behavior can be captured, which in turn affects the distinguishability of two traces.

3.4.9.1 Number of measurements and trace size

We start with FSA attacks, which require only the timing at which the output pin gets faulty and do not require the actual output values. If we do the FSA on the most critical output pin, there are generally fewer measurements needed than for an FSA on a specific output pin, but the size of one trace is the same, one timing value. This is because the signal of a specific output pin is more likely to have either a transition with a comparably larger slack, or no transition at all. FSA on all output pins needs even more measurements and has a larger trace size to store the timing values for all the output pins because it is even more likely that one of the output pin signals has no transition at all or has a transition with a bigger slack.

In the worst case, the measurement demand can be compared to scanning the whole transient circuit output behavior, which requires a lot of measurements to get all the transitions of all output pin signals. However, it might be impossible to scan the whole transient circuit output behavior from the previous state to the actual one if the circuit completely fails after reaching a certain frequency. In this case, only the available part of the transient information should be used. The template size for the template attack on full (available) transient circuit output behavior is the biggest one, because the storage of all output pin values for each measured timestamp is stored.

The most efficient method of the proposed attacks in terms of the number of needed measurements and trace size is the template attack on a specific time point. This attack requires only a single measurement for each trace to get the values of all output pins at a specific time stamp, while providing similar results as the other proposed attacks. The attack on compacted test response signatures and the attack on pass/fail test responses requires several such measurements like the attack on a specific time point, but the values of output pins are not directly stored. So even though it requires more measurements, it does not necessarily have a much larger size of traces and potentially reveals less side-channel information. However, compacted test response signatures still contain some information about the output values and carry a bit more side-channel information according to our results.

If using another type of measurement like our delay line, the circuit is assumed to be running at its original rated clock frequency. Timing resolution, number of timestamps and available range of transient behavior which can be scanned is then dependent on the physical properties of such a delay line. For all proposed attacks, a single measurement is sufficient to generate an appropriate trace. However, it can only work if the signal can be accessed before the register stage.

3.4.9.2 Template size and computational effort

The size of the template for each attack depends on the method it is generated with (exhaustive or ad-hoc) and on the size of a single trace. An exhaustive template will consist of traces for *every possible input transition · every possible key value*, i.e. for our 8 bit SBox setup: $2^8 \cdot (2^8 - 1) \cdot 2^8$ traces. Therefore, the ad-hoc template which consists only of the *number of observed output transitions · every possible key value* (i.e. $30 \cdot 2^8$) traces can be more suitable, as the same set of patterns is usually used for the test.

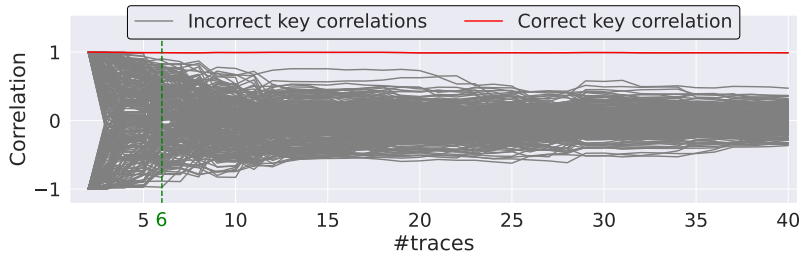
The computational effort for all the attacks consists of preprocessing and formatting the trace data to a suitable form for the correlation calculations if necessary and calculating the correlation values themselves. This depends on the key size, the number of used traces, and the size of a single trace.

3.4.9.3 Amount of side-channel information

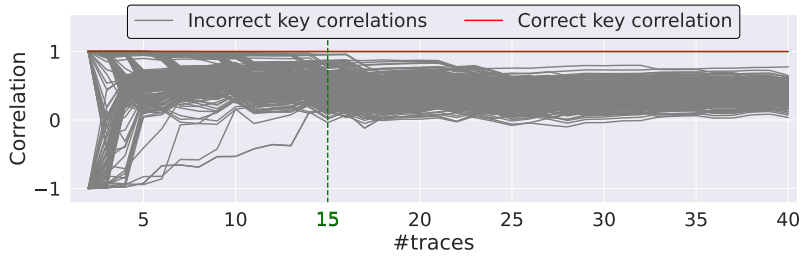
It is important to note here that the number of bits in a trace of transient circuit output behavior does not necessarily define how much side-channel information it contains. The side-channel is actually the information about the signal transitions and its timing. The more fine-grained the resolution, the more precisely transitions can be detected. The longer the observed time interval of the transient behavior, the more transitions can potentially be detected. Theoretically, traces with a longer observed time interval and finer-grained resolution may be more distinguishable from each other and therefore better suited for attacks. Furthermore, such traces of full transient circuit output behavior can also be reduced to a list of transitions with corresponding timings. This reduces the needed storage space, yet calculating the correlation values in such form becomes more complex. However, since the number of transitions and differences in its timings are dependent on particular hardware design and physical properties of the circuit, as well as used input patterns, these define the amount of possible side-channel information in the first place.

3.5 Discussion

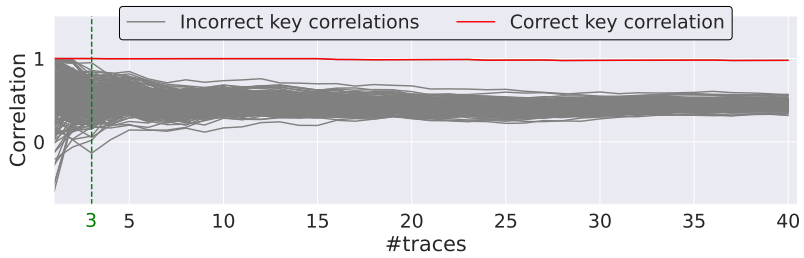
The results presented in this chapter directly show how delay tests and data on transient circuit output behavior can break security. Here, we discuss further aspects of what this can mean for testing and other related fields.



(a) FSA attack on most critical output.



(b) FSA attack on first output pin.

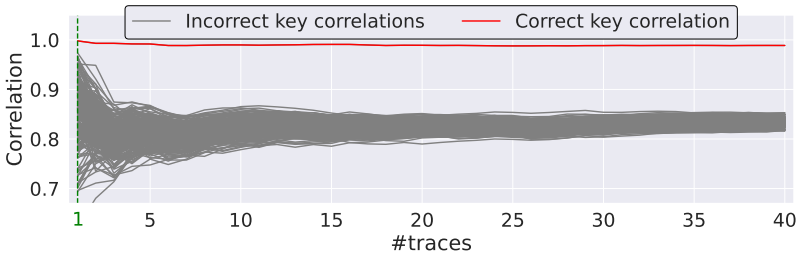


(c) FSA attack on all output pins.

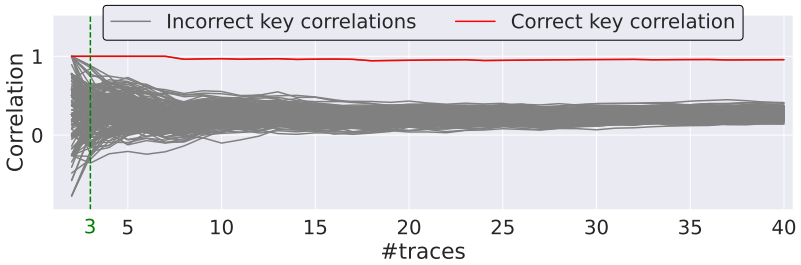
Figure 3.17.: Additional results for attacks with template data gathered at office temperature and attacked traces gathered in climate chamber at 70 °C. Correlation values of the correct key (red) and incorrect keys (grey) depending on the number of used traces. The dashed vertical line indicates the number of traces from which the correct key has higher correlation coefficient than the others.

3.5.1 Security in Testing

The attacks shown here are feasible in various real situations with direct access on delay or Fmax test results. However, due to the characteristics of fault attacks, even if delay test data is encrypted, obfuscated, or compacted, it could still reveal secret information. Test data that is always encrypted in the same way would always give the same result (replay). However, when doing delay testing, a change in test data will already give away the information that something has changed, which in that case must be a fault, in the simple case. Like that, at least our FSA-based attack could fully operate, and reveal secrets just in the same way as before. Even a template can be built since any change in the output would lead to a change in ciphertext as well. Only a completely established secure cryptographic communication channel could prevent that [108], [109], [111], with other consequences on functional safety (i.e. according to ISO 26262 [113]).



(a) Template attack on full transient circuit output behavior trace.



(b) Template attack on specific time point of transient behavior.

Figure 3.18.: Additional results for attacks with template data gathered at office temperature and attacked traces gathered in climate chamber at 70 °C. Correlation values of the correct key (red) and incorrect keys (grey) depending on the number of used traces. The dashed vertical line indicates the number of traces from which the correct key has higher correlation coefficient than the others.

Table 3.2.: Required traces for full key recovery attack. Summary for attacks on traces gathered at different temperature. Typical amount of random traces needed for a successful attack. (*templating* and *victim device* are different devices; template data gathered at 19-24 °C office temperature, data gathered from victim device as noted)

	Victim Device Temperature		
Template Attack data	**0 °C**	**45 °C**	**70 °C**
full transient circuit output behavior trace	2 traces	2 traces	2 traces
specific time point	5 traces	6 traces	6 traces
(FSA) most critical path	25 traces	18 traces	21 traces
(FSA) specific output	29-79 traces	23-80 traces	33-74 traces
(FSA) all outputs	7 traces	6 traces	6 traces
test response signature (MISR)	17 traces	28 traces	24 traces
pass/fail test responses	22 traces	16 traces	26 traces

3.5.2 Attacks on test data

An attacker does not necessarily require direct access to the victim device. For a successful attack, it is sufficient to only acquire some of the test data that contain transient circuit

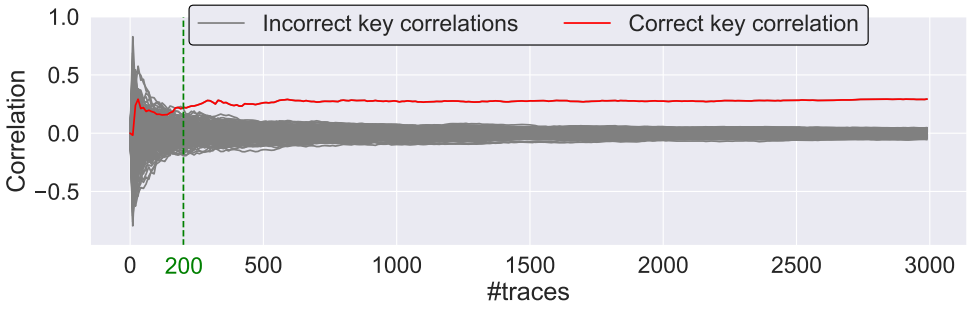


Figure 3.19.: Result for CPA based on transition number as power estimate. Correlation values of the correct key (red) and incorrect keys (grey) depending on the number of used traces. The dashed vertical line indicates the number of traces from which the correct key has higher correlation coefficient than the others.

output behavior information. Dependent on the type of available test data, the attacker can choose a suitable attack like FSA on the most critical output or a Template attack on a specific time point. The data needed for template generation can be simulated; therefore, having access to the circuit netlist can be sufficient without the need of a device copy. There are also other types of attacks thinkable which do not need profiling at all, but require more test data.

3.5.3 Attacks without templating device

If a templating device is not available, a gate level simulation is a very good alternative for creating a template. However, if both are not possible, removing the dependency on a template for the attacks on transient circuit output behavior could be possible with some restrictions. The proposed attacks on full transient behavior or specific time points are rather impossible without a template. However, the data from full transient circuit output behavior can be used to estimate the power for the side-channel analysis. To generate one such power trace we use the transition number from a full transient circuit output behavior trace. We performed multiple experiments using these traces for CPA [22] attack based on the Hamming distance as the power model. This attack was successful starting with about 200 traces (see Figure 3.19).

If the timing characteristics of the circuit are known or can be examined at a similar circuit, the attacker can make a hypothetical function to estimate the timing characteristics of the CUT. This function can later be used for the attack phase instead of the template like in the original FSA attack. Furthermore, an attack on the fault sensitivity can still be possible without knowing any timing characteristics of the circuit by applying a Collision Timing Attack proposed in [139]. Here, the correlation collision attack compares the timing characteristics of two SBox instances to find the linear difference between the corresponding keys, but it is required to observe the fault sensitivity for every possible (input) output.

3.5.4 Testing for Security

The results have shown that searching for SDD can reveal private or secret information stored inside a circuit that can be a security breach. However, we also think that such tests can be used constructively, by providing a security assessment to potential side-channel leakage. They could allow not just to perform testing for different speed-grades, but also *security grades*. The current results are not yet sufficient to fully attribute to this, but require a study on a set of real ICs. Anyhow, related work has already shown that both design (i.e. specific routing) and manufacturing process variation can in fact make a difference to at least power analysis side-channel vulnerability [136].

3.6 Conclusion

It is well known that test infrastructure can be a possible attack vector, such that mechanisms to keep it secure are well established. However, most of the existing attacks have not taken into account that timing behavior or delay test data might still reveal secret information. Especially in the light of more and more advanced cryptographic fault analysis methods, data such as that from SDD might still contain critical information about secret or private data within the circuit, even if they can not be accessed directly through the scan chain. This work subsequently shows that data from small delay or Fmax testing can in fact be security critical, leading to fully revealing information about circuit inputs. Using a newly introduced template attack, just two traces of respectively two encryptions are needed to extract the secret key in the circuit under test. Due to these results, we believe the threat model for potential attacks on test infrastructure needs to be carefully extended to include state-of-the-art fault analysis attacks.

4 Side-Channel Attack on Signal Jitter

The work described in this chapter was first published in "JitSCA: Jitter-based side-channel analysis in picoscale resolution"[2] and is joint work with co-authors Kai Schoos, Dennis R. E. Gnad, and Mehdi B. Tahoori.

Timing side channels [16] typically affect software-implemented cryptography or timing variations in the microarchitecture and are still a practical threat to various systems today [140]–[145]. Power analysis [17] and electromagnetic attacks often no longer need dedicated measurement equipment, and have been shown feasible from inside the same chip [30], [31], [47], [146], [147], the same power domain [48], or close proximity [49]–[51]. When taking a closer look, these attacks cannot be categorized separately, as already Simple Power Analysis (SPA) is basically a timing attack through power measurements [17]. Vice versa, a new range of side-channel attacks observes very fine timing differences caused by physical variations as an estimate of power consumption in the victim [1], [30], [48], [146], [148].

These *timing-based power analysis attacks* typically re-use existing hardware components and reconfigure or use them in a way to be sensitive to power or voltage variations on the device itself [146]–[148]. When an attacker has access to these components, they can thus perform power analysis attacks on other components in the system with the same power domain. Among these, delay line-based sensors have been the most researched, and have been shown sensitive enough to voltage fluctuations not only from the same SoC [30], [148], but also from other components connected to the same power supply [48], [149]. Nevertheless, all of these attacks still work in the same power supply domain, where in general galvanic isolation can improve security somewhat [150], [151]. Timing differences that can be measured from another system were so far leveraged for attacks in a more classical way, when the respective cryptographic implementation was not constant time [140]. What has been shown is that timing jitter in a Controller Area Network (CAN) bus signal can be used to identify a hardware device, due to the respective *manufacturing process variations* of the device [152].

What has not been shown so far, is that minuscule timing differences such as signal jitter can show data-dependent *runtime variations* sufficiently for side-channel attacks. Furthermore, all the mentioned works stay in the same power domain and thus cannot differentiate whether the measured side-channel leakage is from the signal that is measured (i.e. the clock) or the actual sensor being sensitive to the respective physical variations such as voltage [30], [48], [146], [148], [153].

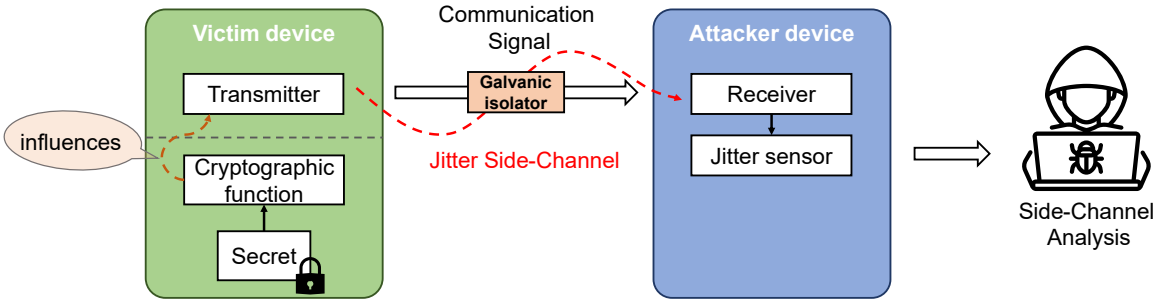


Figure 4.1.: Generic Adversary Model used in this work.

This chapter addresses those points. We show that signal jitter contains enough side-channel information for a key recovery attack, which is dependent on the physical variations in the transmitter of that signal. We can reject that those variations come from direct electrical coupling, since we also show our attack to work with galvanic isolation. To demonstrate that, our experiments are first performed on a single FPGA platform for reference, where our design consists of a victim that is transmitting a clock as its signal, and an attacker that receives this signal, which is essentially reproducing on-chip power analysis. We gradually spread out this design to two FPGAs, then a communication through HDMI, and finally a galvanically isolated HDMI signal. In summary, we make the following contributions:

- We show that signal jitter is a new side-channel attack vector, which we can clearly separate from direct power/voltage side channels.
- Escalating timing-based power analysis attacks from being performed within systems supplied by the same power supply to galvanically isolated communication between two systems.
- By carefully designed experiments, we clearly differentiate between the power leakage observed *inside* an FPGA-based delay sensor, and the *outside* timing leakage from the signal jitter that is measured using the sensor of the adversary.
- We show that our experiments are generic enough to be performed with two FPGA systems from different vendors and a galvanically isolated HDMI link between the boards.

The impact of this work is beyond our results and implies that many systems that were assumed to be connected securely are suspect to this new type of side-channel leakage. Many mission-critical systems in medical and military applications use galvanically isolated device-to-device communication or enforce unidirectional communication [154], [155], but also consumer-oriented or generic networking devices are at risk.

In the remaining paper, we will first summarize background and related work in Section 4.1 and give generic adversarial models for the experiments performed later. In Section 4.2 we will explain how the individual components of information leakage are measured. Our experimental setups are described in Section 4.3. In Section 4.4 we show our results with an interpretation and discussion in Section 4.5. Finally, Section 4.6 concludes the paper.

4.1 Preliminaries

4.1.1 Adversarial Models

The results from this paper can apply to various scenarios. However, in all cases the attacker has access to a communication signal coming from the victim, while the victim is performing sensitive computations such as cryptography, as shown in Figure 4.1. These computations affect the transmitted signal, and even if it is galvanically isolated, an adversary can measure jitter, tiny timing deviations from true periodicity, of the received signal. This information can then be used for side-channel analysis to recover the secret information.

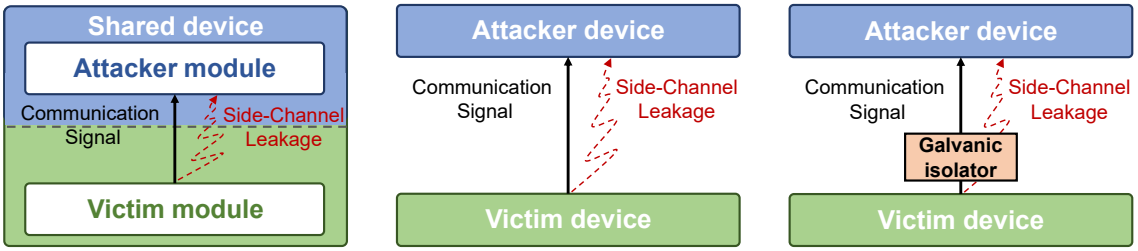
Additionally, we assume three scenarios of the generic adversarial models shown in Figure 4.2, which are later used in our experimental setups. The on-chip case Figure 4.2(a) assumes that the attacker and victim are on the same chip, and the attacker has access to a communication signal from the victim. Both, the attacker and the victim share the same power supply, which is also addressed in previous publications [30], [48], [146], [148], [149]. This scenario can occur inside SoCs or cloud environments, where the tenants have access to different parts of the hardware. This model is mainly used as a baseline comparison in this paper.

For the inter-device case Figure 4.2(b), attacker and victim are on different devices. They do not share a power supply but are still connected by the same ground. The attacker receives a communication signal from the victim, which carries the side-channel information. This is the most common case and includes scenarios like an attacker having access to the interface of a digital clock or only having access to the communication I/O signals of an otherwise tamper-resistant device operating on sensitive data.

Finally, for the isolated inter-device case Figure 4.2(c), the two devices are only connected by a galvanically isolated digital communication signal. Both devices do not share any common electrical reference. This setup can be found in devices that transmit data via optical links or galvanically isolated connections, as they are found, for example, in high-speed networking devices or in various mission-critical appliances with respect to safety and security [154].

4.1.2 Related Side-channel Attacks

Timing-based side-channel attacks typically use variations in the runtime of software, which can leak information secrets about the data that is being processed [16]. Even though such attacks have been known for over two decades, timing attacks can still be real-world problems, as recently shown by breaking firmware-based TPMs [141]. Furthermore, modern CPU architectures use various best-effort optimizations that lead to inconsistent timing that gets increasingly exploited [143]–[145], [147], especially since the introduction of Spectre attacks [142].



(a) On-Chip (b) Inter-Device (c) Isolated Inter-Device

Figure 4.2.: Adversary Models. (a) On-chip attack, where victim and attacker reside on the same chip. (b) Inter-device attack, where victim and attacker share ground and communication signals. (c) Galvanically isolated inter-device attack, where victim and attacker only share a galvanically isolated communication signal.

Power analysis attacks are different in a way that they are usually harder to prevent from the software implementation level compared to timing. Thus, cryptographic accelerators often leak information through data-dependent power consumption, if no specific attempts are made to prevent it [156]. Since the introduction of DPA by Kocher et al. [17], more and more attacks have shown the security risk from allowing unprivileged power measurements [157], for which countermeasures of various types are actively developed [24], [151], [156]. However, as power analysis attacks analyze the power consumption over time, they implicitly and sometimes explicitly consider timing as well [17].

More recently, various types of very fast timing measurements have also been shown to allow indirect measurements of power, which we would call *timing-based power side channel attacks*. Inside FPGAs, TDCs have been used to measure transistor delay, which is among other effects affected by the voltage of the entire FPGA chip, and can be used for power analysis attacks [30], [146], [153]. Similarly, generic delay lines in SoCs were also shown to allow side-channel attacks [148], which extends such attacks to much more potential systems. In [1] chip testing methods are used to directly see the transient circuit behavior that would later cause differences in power consumption that can be used for attacks. Extending the FPGA-based attacks beyond a single chip, it was shown that even other devices connected to the same power supply could be measured [48], [149]. Timing measurements inside a processor can also be used as a proxy for the power consumption of the respective processor, when the scheduling of Dynamic Voltage and Frequency Scaling (DVFS) is being monitored, leading to a successful key recovery attack [147]. These works looked into self-measuring the power consumption of the respective chip, while in this paper, we will look into the timing of the signals received from *another* device, where the jitter in the signal will be a proxy for the power consumption of the transmitting device.

Regarding galvanic isolation, it was shown that output port pins of a victim chip with a constant value can be connected to an optocoupler, and when the receiving side is measured, the analog variations from the victim are still observable in the coupled side [150], but increasing the required traces for a successful power analysis attack by about 4500×. In [151] an ASIC was manufactured with a galvanically isolated AES module integrated; they

stopped measuring after about 600× as many traces than for their non-isolated comparison and could not launch a successful attack with that.

Considering on-chip power side-channels in related work, they were not just exploited through delay-based sensors in FPGAs [30], [146]. It was also shown that the influence from digital logic can affect analog components in the same SoC, which can influence a Digital-to-Analog Converter (DAC) and similarly it can even end up in a wireless signal [50]. Other works have shown that on the system itself, the noise of an Analog-to-Digital Converter (ADC) can be used for power analysis attacks, sufficient to break through privilege levels [31], [47], [158]. These works acquire side-channel information in the same power domain, by measuring analog properties, such as time and magnitude of a signal. The typical time resolution is in the range of clock cycles of the victim system, i.e. $10 - 100$ ns, with a typical resolution of $6 - 12$ bit. In comparison, we only measure a discrete digital 1-bit signal from which also a discrete (clocked) time behavior is expected. However, because we measure at a much higher time resolution than previous work (12 ps), analog timing properties in the form of jitter become visible, which we exploit as a side channel.

4.1.3 Power model and Leakage Assessment

In this work, we execute a CPA attack [22], and correlate the actual jitter measurements as a power estimate with a model. The used power model is slightly different from classical CPA attack and targets a single bit instead of a whole byte, as previous works in FPGAs have done successfully [146]. For the attack on the last encryption round of AES, we model power with $p_m = sbox^{-1}(k_{guess} \oplus c_i) \wedge 2^b$, where k_{guess} is the key byte guess, c_i is the corresponding ciphertext byte and b is the bit selected for the attack. The attack to recover one key byte is successful if at least one of its bits has a higher absolute correlation value than the other key guesses. In this work, we will report the required amount of traces for the correct key guess to be the point after which no other key will correlate more than the correct one. For our experiments, we use the FIPS AES test cipher key *2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c* [159].

To generally show the presence of first order statistical leakage in our experiments, we analyze two sets of traces as introduced in TVLA [160]. The first set is measured when computing on a fixed input value, and the second set when computing on various random input values. Then, a Welch's t-test is performed to statistically compare the two sets. When the two sets are indistinguishable, no leakage is assumed.

Due to the high number of samples K per trace, the t-test cannot be evaluated against a single threshold value TH_t. As the number of samples per trace grows, the chance of having certain samples randomly surpassing a fixed threshold grows with it. In fact, for traces with 1 million sample points, 99.9% of leakage-free devices are classified as leaky under a threshold of $TH_t = 4.5$, which is classically used [161]. The threshold for a given

significance level can be found by interpreting all of the t-tests as a mini-p procedure. The threshold is then computed by:

$$\alpha_{TH} = 1 - (1 - \alpha)^{\frac{1}{K}} \tag{4.1}$$

$$TH_t = CDF^{-1}_{\mathcal{N}(0,1)}(1 - \alpha_{TH}/2) \tag{4.2}$$

With α being the desired significance level and K the number of samples per trace. Theoretically, we would need to use the Cumulative Distribution Function (CDF) for the t-distribution here. However, as the CDF for the t-distribution converges toward the CDF for the standard normal distribution $\mathcal{N}(0,1)$ for high degrees of freedom, the latter can be used, as this boundary condition holds in our case. This kind of evaluation is used for what is called the *TVLA total-plots* in this work.

A different evaluation method based on Higher Criticism (HC) is used for what is called the *TVLA progress-plots* in this work [161]. This method takes into account the distribution of the t-values and thus gains more detection power for devices with some countermeasures in place [161]. To carry out this procedure, a HC statistic is computed for each of the K t-values where K corresponds to the number of time steps per trace. First, the t-values are transformed into their respective p-values by computing the survival function $1 - CDF_{\mathcal{N}(0,1)}(t)$. Again, the CDF for the standard normal distribution can be used here because of how the t-distribution converges towards $\mathcal{N}(0,1)$ for high degrees of freedom. The p-values are then sorted in ascending order, before the HC estimator is computed as follows:

$$\widehat{HC}_{K,i} = \frac{\sqrt{K}(\frac{i}{K} - p_i)}{\sqrt{p_i(1-p_i)}} \qquad \text{for i=1,...,K} \tag{4.3}$$

$$\widehat{HC}_{K,max} = \max_{1 \le i \le \frac{K}{2}} \widehat{HC}_{K,i} \tag{4.4}$$

Here, K HC estimator values $\widehat{HC}_{K,i}$ are computed. These are the individual HC-values for a given trace of length K, and p are the p-values received from the t-values, sorted in ascending order. $\widehat{HC}_{K,max}$ is the final statistic that will be thresholded to find out whether or not the signal is leaking. The number of traces N does not have any effect on the computation. This is due to the fact that for a large number of degrees of freedom (df), the t-distribution simply becomes a standard normal distribution. Even for the smallest amount of traces analyzed (1000), df is large enough for this boundary condition to be met.

In order to find the threshold $b^{HC}_{K,\alpha}$, a Monte Carlo simulation is run. For the null hypothesis, the p-values follow a uniform distribution $p \sim U(0,1)$. Thus, for the simulation, $\widehat{HC}_{K,max,null}$ is computed 1,000,000 times for K p-values sampled from $U(0,1)$ in order to estimate the distribution for $\widehat{HC}_{K,max,null}$. The threshold $b^{HC}_{K,\alpha}$ is then defined as the $(1-\alpha)$ quantile of the $\widehat{HC}_{K,max}$ distribution, with α being the significance level. The signal is said to be leaking if $\widehat{HC}_{K,i} \ge b^{HC}_{K,\alpha}$.

4.1.4 Signal to Noise Ratio (SNR)

The Signal to Noise Ratio (SNR) is used in order to compare the amount of leakage between different conditions. First, the power model is computed for one correct key byte for all the traces. After that, the traces are grouped according to the value of the power model. In our case of the bitwise power model, the traces fall into one of two groups, where the power model either evaluates to a '0' or a '1'. For each group $g \in [0, 1]$ and each time step $k \in [1, ..., K]$, the signal $Q_{g,k}$ is computed as the arithmetic mean over the traces that are part of that group $\mathcal{T}^g$.

$$Q_{g,k} = mean(\mathcal{T}_k^g) \tag{4.5}$$

The noise $v_{n,k}$ of each of the NxK measurements is determined by subtracting the signal of the group of the n-th trace $Q_{g,k}$ from the original measurement $\mathcal{T}_{n,k}$:

$$v_{n,k} = \mathcal{T}_{n,k} - Q_{g,k} \tag{4.6}$$

Finally, the Signal to Noise Ratio (SNR) for each timestep SNR_k is determined by computing the variance across all the groups for the signal and the noise and taking their quotient:

$$SNR_k = \frac{Var(Q_k)}{Var(v_k)} \tag{4.7}$$

Where Q_k are the signal values for all the groups for time step k and v_k are all of the noise values for time step k.

4.1.5 High-Speed Timing Measurements with Delay Lines

One possible way to implement a TDC is a delay line. TDCs were originally used in single timing measurements for physical experiments that required a high resolution [162], and are also used in chip testing [1], [163]. In an abstract way, a *Start* and a *Stop* signal control the TDC, which measures the time between the signals, after which its data gets collected to measure that timing difference [162]. When tapped delay lines are used, memory elements are added between a long line of buffers or inverters. The start signal is fed into the delay line, while the stop signal controls the memory elements (registers or latches). Like that, the progression of the start signal is an indication of how much time has passed between the signals.

By using the same signal for both start and stop, not the delay of the signal is measured, but instead it is an indicator of the power consumption of the device itself, which can be used for power analysis attacks [48], [146], [148], [153]. Our variation of that sensor is shown in Figure 4.3. Delay lines or TDCs are available in many devices already [148], [163] such that no external measurement device is needed, and for flexibility and experimental purposes can also be synthesized into FPGA fabric [48], [146].

In detail, these TDCs are made up of two general parts: A delay line and the storage registers. When measuring an estimate for power, the clock is fed into the delay line and

into the clock connections for the storage registers. For ideal components, the input signal (i.e. the clock signal in this case) would instantly propagate to the end of the delay line. However, in the real world, the time that the clock edge takes to travel through the delay line is non-zero. The storage registers all record the value of their corresponding latch at the moment the clock edge reaches their clock input, freezing the state of the delay line at that moment. Typically, the delay line is fine-tuned manually by changing its length, such that the previous clock edge is being propagated through the delay line when the storage registers are clocked, thereby sampling the delay between the previous (negative) clock edge and the current (positive) clock edge. The result is a string of '1's followed by a string of '0's, marking the clock edge. The point in the string where the transition from 1's to 0's happen thus marks the length of the last negative pulse. The variation of this value was shown to be an estimator for the variation of the voltage level of the system, which can be used in power analysis attacks [146], [148], [153].

4.1.6 Jitter and its Measurement

Typically, signal jitter is categorized into bounded and unbounded jitter, where bounded jitter is further broken down into correlated and uncorrelated jitter [164]. Data Dependent jitter (DDj) is considered to be correlated to the transmitted data, but voltage fluctuations of the transmitter are typically addressed as a form of random or uncorrelated jitter.

Others have already explored analyzing timing variations of communication channels in the form of jitter, but not to extract secret data. More specifically, in [152] an identification approach is shown that measures jitter on an automotive CAN bus by using an FPGA-based TDC sensor. This jitter is used to uniquely identify the respective transmitter of the message by detecting its typical jitter characteristics, influenced by (systematic) manufacturing process variations. The respective input to their measurements comes from a CAN transceiver module and from there is forwarded to a coarse time-sampling circuit. The coarse time sampler measures multiple CAN messages, which are then forwarded to a TDC circuit that measures the period between messages, i.e. the rise and fall times of the CAN signal. In the end, they report a minimum delay resolution of $219ps$, which is sufficient to measure the slow 500 kbps CAN bus signal with comparatively high jitter in [152]. However, that resolution would not be sufficient for this work, where jitter was observed to be usually less than $100ps$.

Alternatives to this measurement method would be to use an external spectrum analyzer and continuously measure the phase noise of the system, where care should be taken that the frequency of the local oscillator of the device is close to the frequency of the signal under test. Furthermore, repetitive measurements would be required. Digital oscilloscopes, on the other hand, may have problems measuring such high-resolution phase variation out of the box, since $11.5ps$ resolution relates to about 87 GHz.

In this work, we will use a delay line TDC FPGA design explained in Section 4.1.5, that has been reported with a timing resolution of about $11.5ps$ in one of the same FPGA platforms that we use [1]. It uses the delay from the input to the sampling registers of the delay

line to catch a single period of the signal. The aim is also different from [152] in that we do not want to measure jitter that can be reproduced per transmitter, but jitter that changes its characteristic depending on the data being processed inside the transmitter. To measure jitter for side-channel analysis, we implement a configurable delay line based TDC following the design from [1], which adds flexibility regarding the time range that we want to measure. Further changes are described in Section 4.2.1.

4.2 Methodology

In this work, we measure the timing variations of a (clock) signal transmitted from a victim device to the attacker device, also known as jitter. An attacker receives this signal, and measures this signal's jitter. While the transmission happens, the victim is also running a cryptographic accelerator, which we assume causes minor voltage fluctuations in the entire power domain of the victim circuit. These voltage fluctuations also have an impact on the cycle-by-cycle jitter of the transmitted signal, sufficient for side channel-based key recovery attacks.

The attacker module aims to measure this signal jitter and derive the victim's power consumption and by that secret keys from the cryptographic operations. With an FPGA-based TDC, the attacker will measure the delay between two consecutive edges of the received signal, which can also be run continuously to get a reading on every change of the data signal. The variation of this delay is then a unit-less value of the jitter that can be used in side-channel analysis, where in this paper CPA and TVLA are used.

TDC sensors are also directly sensitive to voltage and temperature variations. This relation makes it possible to use timing variations of a measured (clock) signal with a TDC sensor as a power estimate for side-channel analysis. In related work, the TDC sensor is usually in the same power domain as the victim, and the measured (clock) signal itself as well as the delay line components are affected by the voltage fluctuations in this power domain. Therefore, an adversary can measure voltage fluctuations caused by a running victim cryptographic circuit.

This section explains how we measure signal jitter with delay line-based TDCs, as well as how we designed our experiments to allow a distinction between power in the PDN supplying the delay line versus the estimate we get through jitter from the victim, potentially even when the systems are galvanically isolated and no direct leakage through the PDN is possible.

4.2.1 Measuring Jitter for Side-Channel Analysis

As introduced earlier, we use delay line-based TDCs to measure signal jitter. Since TDC sensors are also sensitive to voltage and temperature variations, they are usable in power-based side-channel attacks, which we will separate based on the experimental setup.

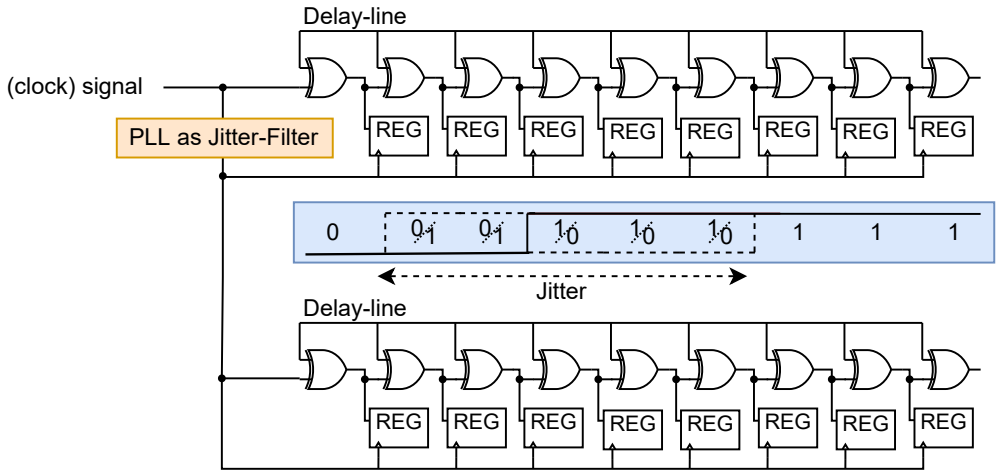
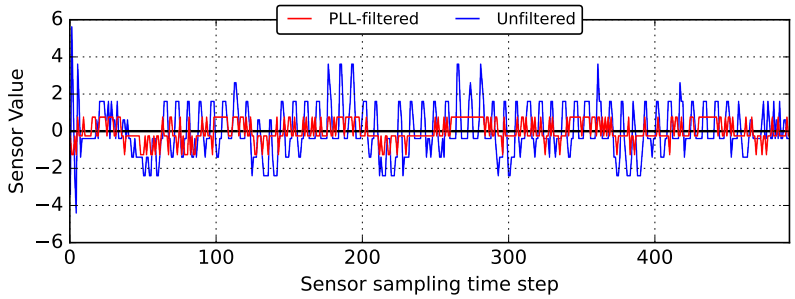


Figure 4.3.: Delay line-based TDC sensor to measure voltage from variations in the speed of the XOR cells as well as from the clock signal. A clock enters the delay line while they are sampled by a previous edge from the same clock signal, which will include clock jitter. This is our interpretation of previously published FPGA-based voltage sensors [146].

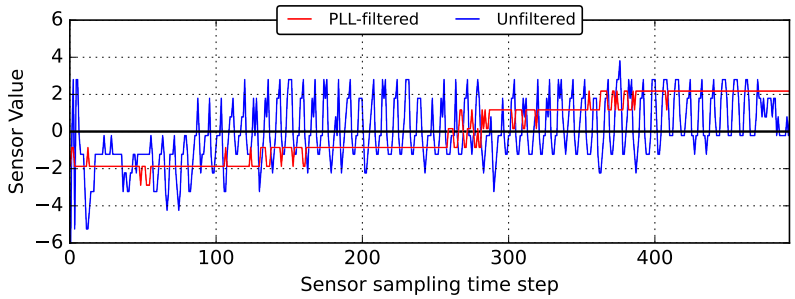
The basic structure of the used TDC sensors is illustrated in Figure 4.3. The same (clock) signal is propagated through the delay line and is used to trigger the storage registers. With a suitable variation of the delay line length, which can be configured manually for every experiment, relative time differences from the actual rising edge to the falling edge of the previous clock cycle can be measured. The measured time window that a TDC is measuring is generally fixed. For clock signals, every rising edge is preceded by a falling edge in a relatively short time window for which the TDC is configured. This is not true for general data signals. Therefore, when measuring jitter on a data signal, some traces may become useless and need to be sorted out while preprocessing the data.

The (clock) signal transition itself is stored in the storage registers as a sequence of 0's followed by a sequence of 1's. This represents a relative time point of the actual (clock) signal transition in the previous clock cycle, with a discrete resolution of about 11.5 picoseconds for one of the used boards platforms (PYNQ-Z1) [1], but unknown for our other ULX3S platform. To receive the final sensor value, the values of the storage registers are summed up. This quantity reflects the duration of the latest negative clock pulse, which is the time of the nominal clock period affected by jitter. For that, we show exemplary measurement traces in Figure 4.4 for both of the systems used in our experiments.

For our experiments, we use two delay line based TDC sensors in parallel where the second sensor acts as a reference measurement, as shown in Figure 4.3. While the first sensor measures the jitter in the signal as it is, for the second one we try to filter jitter first. For this, the signal is passed through a PLL with an output clock of the same frequency, since PLLs have the property of filtering out jitter from the source clock, while maintaining a constant phase relationship to it. Thus, if we use the output of the PLL as an input signal to the TDC, this measurement should contain almost no jitter (and no side-channel leakage) from the original clock.



(a) Single trace captured using PYNQ-Z1



(b) Single trace captured using ULX3S

Figure 4.4.: Exemplary Traces of TDC measurements used in the experiments. The blue lines show traces captured while measuring the unfiltered signal. The red lines show traces captured while measuring a signal filtered using a PLL. The red and blue traces shown for each board are captured in parallel.

The PLL-filtered and unfiltered measurements are both included in Figure 4.4 as red and blue plot respectively, showing that the PLL in fact reduces jitter. To what extent that translates to side-channel attack success will be explored later. Filtering the jitter with a PLL can still be insufficient and include side-channel information, especially if the PLL itself is in the same PDN as the victim. Please note, in all cases, using the PLL as a filter only acts as an experimental vehicle applied by the attacker. It cannot work as a countermeasure, which is why we also need to use galvanic isolation in our experiments, which we discuss later.

4.2.2 Separation of Leakage from Voltage Influence on the TDC

We hypothesize in this paper that the signal-jitter carries a lot of the side-channel leakage, and thus we need to show that the leakage is only in part due to the electrical connection between the victim and attacker. In literature, the voltage fluctuations in the PDN that supplies the TDC are described as the main influence on the observed variations at its output [146], [148]. However, to our knowledge, these voltage variations have never been experimentally separated from clock jitter thus far.

We consider that an electrical signal and required ground-connection will still have a minor influence on the attacker's PDN that supplies the TDC, and thus we need to be

able to reject this hypothesis of a still-existing power coupling which can be reached with galvanic isolation. Furthermore, we try to filter jitter out of the signal and through that evaluate if some leakage can still be measured.

In order to prove our assumptions, multiple experiments can be used to confirm each other. Here we explain the ideas behind the experiments, which are later detailed in the experimental setup:

- Single board in which victim and attack circuit share the same power supply and clock. The attacker measures the clock with a TDC sensor, reproducing related work such as [146] and acting as a comparison baseline. This will be the **On-Chip** experiments (Section 4.3.1).
- Two boards that have a communication link. The victim operates AES while he is sending an independent message to the attacker through the communication link. A successful attack that uses the signal jitter of the communication link as a side channel will be a necessary precondition to confirm that signal jitter alone is a sufficient side channel. This will be done in the **Wire and HDMI** experiments (Section 4.3.2 and Section 4.3.3).
- When performing the previous experiment through a galvanically isolated communication channel, it will finally confirm our main hypothesis, this is done in the **Isolated HDMI** experiment (Section 4.3.3).
- To cross-check that our galvanic isolation does not allow for power side-channel attacks, we filter the signal jitter with a PLL on the attacker side and try to use this for attacks. While it is most certainly not a perfect filter, an unsuccessful attack can show that the most significant leakage was acquired through the signal jitter and not any other part of the TDC.

As a check, we will additionally perform each attack with PLL-filtering in all of the experiments, using the described two delay lines in Figure 4.3, to get a better picture of the filtering capabilities. However, the filtering with a PLL only acts as an experimental vehicle and cannot be a countermeasure. It will always be done on the side of the attacker. If it is applied in the same power domain as the victim, it would again be influenced by power supply leakage. Furthermore, it can only filter clock signals and no irregular communication signals.

4.2.3 Attack Flow

For the attack flow to be possible, the malicious actor must be able to prompt the victim device to perform a sensitive operation or the victim device itself must repeatedly perform some kind of sensitive operation. During this operation, the attacker FPGA must have access to a communication signal, such as a clock signal or transmitted HDMI signal of the victim. The attacker then repeatedly collects measurement traces. These traces are repeated measurements of the communication signal and are stored together with the artifacts generated by the sensitive operation on a workstation PC connected to the

attacker FPGA. In our case, these artifacts are the ciphertexts computed by the AES module. After collecting enough traces, CPA and TVLA are performed off-line on the workstation PC.

4.3 Experimental Setup

The experiments in this paper are conducted with two different kinds of FPGA boards, the Xilinx PYNQ-Z1 that includes a Zynq SoC with ARM processor and FPGA logic with 53k LUTs, and the Radiona ULX3S, which includes an Espressif ESP32 microcontroller and a Lattice ECP5 85F FPGA with 85k LUTs. However, we do not use the processors but only the FPGAs, which are both directly connected to the respective workstation PC through USB for programming, serial communication, and power. We use two ULX3S boards that we enumerate with #1 and #2 throughout the paper. For most of the experiments, the boards are powered from independent power sources, and in some of them, we use an Analog Devices EVAL-CN0422 evaluation board for galvanic isolation of HDMI signals [165].

The TDCs on the different attacker devices use fast carry-chain elements, which are either Carry4 elements on the PYNQ-Z1 or CCU2C elements on the ULX3S. The version of AES-128 that is used for the experiments is a simple implementation using 4 parallel S-Boxes, where each round is performed in 5 clock cycles. On both platforms, the TDC is operating at 100 MHz, while AES is always clocked at 12.5 MHz. Details on the individual setups are listed in the following subsections. Please note that 12.5 MHz might be considered low, but this does not mean that higher speeds do not leak; we just did not perform the respective experiments and wanted to show the general possibility of leakage.

4.3.1 On-Chip: Attacker and victim on a Single Device

This first experiment is performed on a ULX3S board and replicates and extends the concept of internal power analysis attacks from [146], which we assume includes the effect of voltage fluctuations both on the chip-internal clock generator resulting in clock jitter, as well as the impact on transistor delay inside the delay line used for the measurements. With the results from this experiment, we will have a baseline for the remaining results.

Figure 4.5 shows a block diagram, giving an overview of this setup. Two clocks drive the design. A fast clock of 100 MHz is used for most of the design as well as the delay line sensors, and a slower clock with 12.5 MHz drives the cryptographic module. Both clocks are derived from a 25 MHz crystal oscillator that is mounted on the development board. The main components of the FPGA on the left-hand side are the cryptographic module for computing AES and the voltage sensor. These two modules send their data to the workstation PC on the right-hand side, providing the attacker with the data they need for ciphertext-based CPA.

The plain text that acts as an input for the AES module is generated using a Linear Feedback Shift Register (LFSR) in order to keep the communication overhead small and

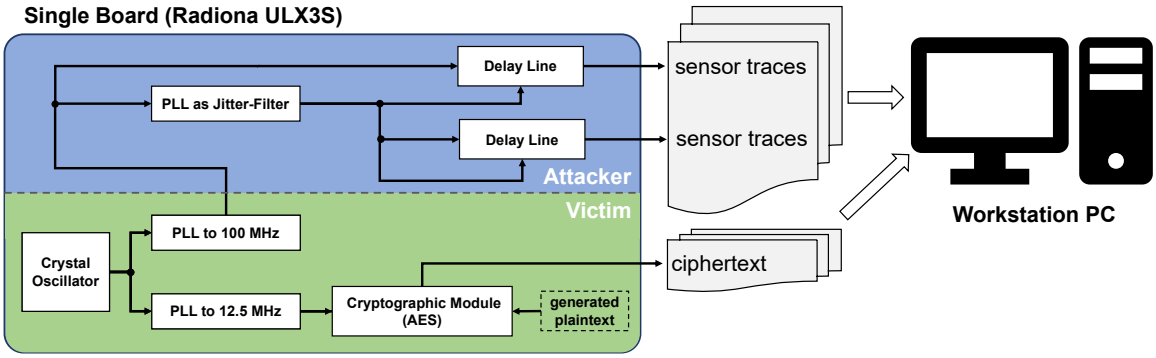


Figure 4.5.: Block diagram for the experimental setup of the On-Chip case.

enable fast trace collection during the experiments. For that same reason, the secret AES key is hard-coded into the device. Care has been taken to not have any overlap between communicating the computed ciphertext back to the workstation PC and voltage measurements, such that the traces can only depend on the actual encryption process, which is needed for fixed vs random testing of TVLA.

As mentioned before, we need to make sure that we are able to differentiate between jitter and power side-channel effects. For this, in parallel to the replication of internal power analysis attacks [146], a second on-chip experiment is conducted that will serve as a control experiment for the experiments that follow. In this experiment, the input signal to the TDC is first filtered using a PLL. This filtering reduces the amount of jitter in the measured signal. Because the PLL is driven by the same PDN as the rest of the victim, this measurement should still show some amount of side-channel leakage. This is in contrast with the following experiments on multiple devices, where we expect that the measurements of the unfiltered signal do show side-channel leakage while the measurements of the filtered signal do not.

4.3.2 Wire: Attacker and victim connected via Twisted Pair Cable

The setup from the on-chip experiments was expanded to the cross-device condition, essentially by distributing the same design across two FPGAs. This setup has three different variations using different boards, which can be seen in Table 4.1. This time there is a clear distinction between the attacker and the victim, in contrast to the on-chip condition. One board (the victim) executes a design that simply computes a ciphertext from a random plaintext and hard-coded key using AES. The plaintext is generated by the LFSR module, just as for the on-chip condition, and also the key is identical to the on-chip experiments. Several control signals are shared between the attacker and victim boards in order to streamline and synchronize the measurement process.

Figure 4.6 displays the distinction between attacker and victim for the cross-device case and shows how different clocks are distributed across the devices. For these experiments, the devices do not share a power connection, but a ground connection. To keep the signal integrity high, we connect ground and the signal wire as a twisted pair from victim to

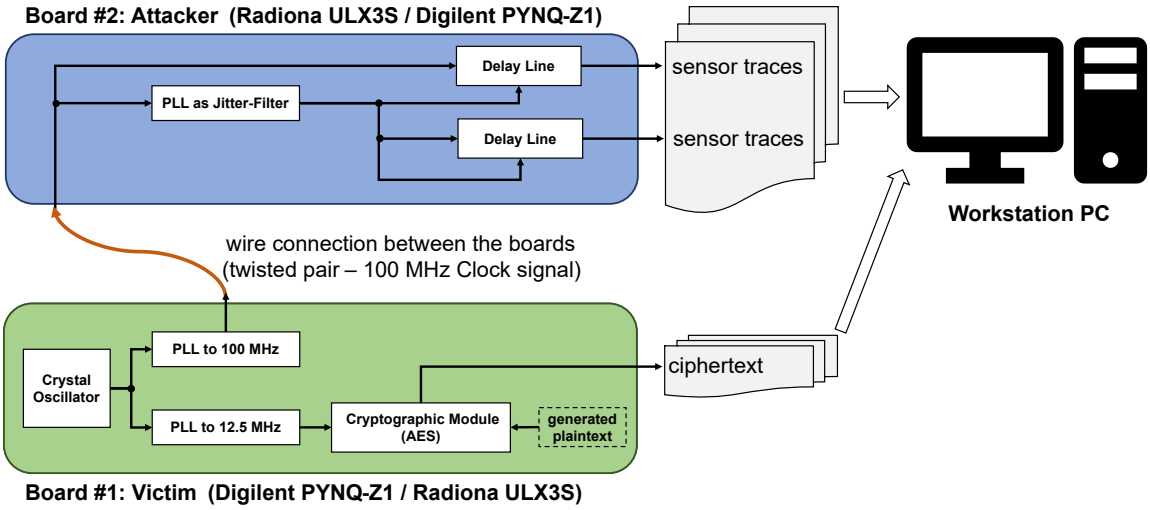


Figure 4.6.: Block diagram for the experimental setup of the cross-device Wire case.

attacker. Both devices are connected to the workstation PC in order to transfer traces (attacker) as well as the ciphertext (victim) to enable the ciphertext-based CPA. The victim device is powered using an external power supply, while the attacker is powered directly via the laptop's USB port.

4.3.3 (Isolated) HDMI: Attacker and victim connected via HDMI cable

For the second cross-device experiment, instead of using a single clock line, we send an NRZS encoded signal through HDMI, which is the only communication link between attacker and victim. NRZS is a form of Non-return-to-zero inverted (NRZI) encoding. In NRZI, one of the binary values ('0' or '1') is encoded by inverting the data signal, while the other value is encoded by keeping it constant. The *S* in NRZS means that the inversion happens on a logic '0' while the signal stays constant for a logic '1'. NRZI is used in various technologies today, like USB and fiber optic communication [166]. The HDMI link is connected to a pair of Low-Voltage Differential Signaling (LVDS)-pins, which is terminated on the attacker side by differential input buffers. The ULX3S is equipped with an General Purpose Differential Interface (GPDI)-interface to support HDMI, which can only be used as an output. Because of that, we only use the PYNQ-Z1 Board as an attacker with HDMI.

The rest of the setup is similar to that of the twisted pair connection. However, the attack flow is changed to maintain complete galvanic isolation and address the limitations of unidirectional communication. The victim repeatedly computes AES on an internally generated random plaintext message. Before starting the encryption process, it sends an NRZS-encoded magic word through the HDMI link, which triggers the attacker to start the measurement process. During the time AES is computing on the victim, it only sends NRZS-encoded 0's through the HDMI link, which is essentially a clock at 50 MHz. After

the encryption process, the victim sends another magic word followed by the ciphertext to the attacker device, which then sends the measured jitter data (trace) and the associated ciphertext to the workstation PC for further side-channel analysis.

In detail, the victim first sends the magic word `0xBEEF`, then sends a series of 0's until AES finishes (always exactly the same constant time), and then `0xFACE` followed by the ciphertext. After a fixed amount of idle waiting during which more 0's are sent, this is repeated with `0xCAFE`, 0's, `0xFACE`, ciphertext. This means it alternates between `0xBEEF` and `0xCAFE` as the starting magic word. The attacker has no handshaking possibility and needs to properly listen to these messages of the victim to record the needed data for performing CPA or TVLA.

To show our initial claim to be true, we additionally use an HDMI galvanic isolation module EVAL-CN0422 from Analog Devices [165], which uses multiple isolator ICs. For us, the ADN4654 IC is relevant, that is used for the main HDMI TDMS signals. It is rated as an isolator up to 3.75 kV RMS, compliant with TIA/EIA-644-A and minimum 50-year continuous AC and DC working voltages of 424 V and 536 V respectively. It also has a jitter rating that is described to not include stimulus jitter, i.e. it is added on top of existing signal jitter, with unbounded jitter specified with 2.6-4.8 ps RMS and bounded jitter of 50-116 ps. The ADN4654 isolation mechanism is based on integrated transformer coils. The signal is encoded into small $\approx$ 1 ns pulses, which on the other side sets or resets a bistable decoder bit to indicate input transitions.

When connecting one HDMI cable for the sender and one for the receiver on the EVAL-CN0422 board, this setup can be used just like a standard HDMI cable. It guarantees galvanic isolation for both the differential and the single-ended HDMI signals. So instead of using a single HDMI cable between victim and attacker, we now use two HDMI cables, one from ULX3S to the EVAL-CN0422, and another one from there to the PYNQ-Z1 board.

To eliminate even more potential information side-channels, each of the attacker system, isolation module and victim device runs from their own respective battery, making it impossible to have leakage through mains power [167]. In addition, the victim and the isolation module are placed inside an EM-isolated thermal chamber (Weisstechnik LabEvent T/210/40/EMC [168]). Since it isolates very well, we cannot keep it off, since the temperature would increase over the time of collecting traces (about 4 hours). Thus, we run it at 22°C, which was also close to the room temperature. Otherwise, the experimental setup stays identical. This final setup is summarized in Figure 4.7.

4.3.4 Summary of Setups

We summarize our experimental setups in Table 4.1, listing the respective platforms and connection mechanisms. We differentiate between **1.** on-chip attacks, **2.** attacks through a normal wire, i.e. a single-ended twisted pair cable with a signal/ground pair on multiple platforms, and **3.** attacks based on a differential NRZS-encoded signal through an HDMI cable, also with additional isolation through an isolator module. For all of the setups, we will measure with the two sensor variants described in Figure 4.3.

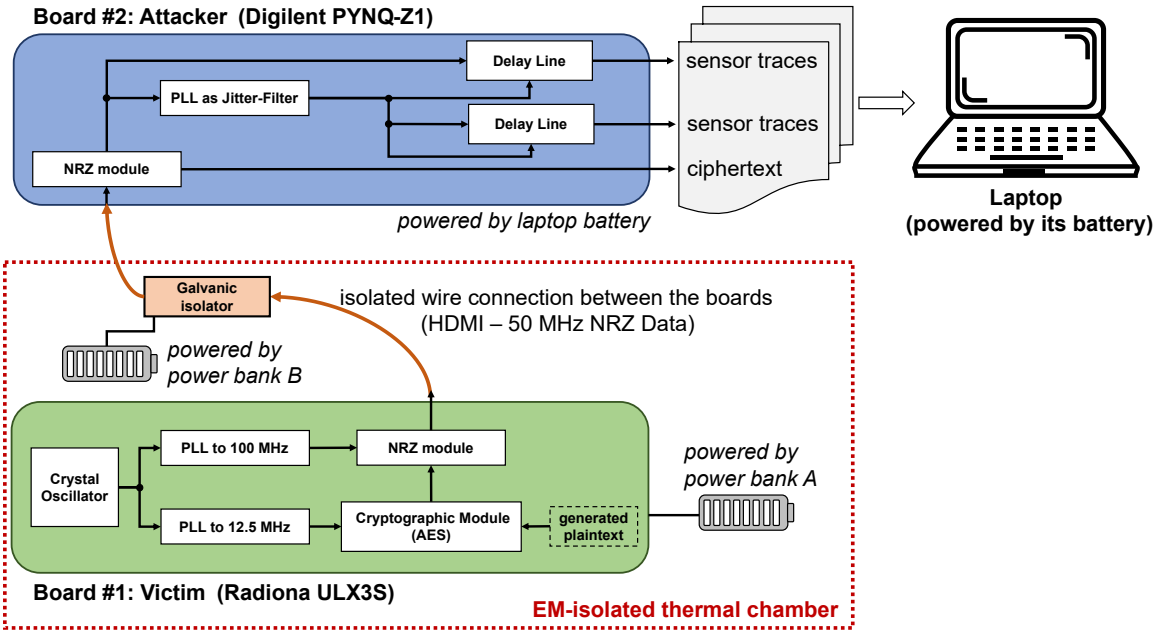


Figure 4.7.: Block diagram for the experimental setup of the cross-device HDMI case, galvanically isolated with the EVAL-CN0422 [165]. All devices are battery powered, so the attacker and victim do not share a common power domain. In addition, the victim device and galvanic isolator are housed in an EM-isolated thermal chamber.

Table 4.1.: Overview of the attacker and victim platforms used for the various experiments.

Basic Setup	Victim / Attacker Platform	Connection	Signal
On-Chip	ULX3S #1	*on-chip*	Clock
Wire	ULX3S #1 / ULX3S #2	Twisted Pair	Clock
Wire	ULX3S #1 / PYNQ-Z1	Twisted Pair	Clock
Wire	PYNQ-Z1 / ULX3S #1	Twisted Pair	Clock
HDMI	ULX3S #1 / PYNQ-Z1	HDMI cable	NRZ all-0's
HDMI	ULX3S #1 / PYNQ-Z1	Isolated HDMI cables	NRZ all-0's

4.4 Results

In the same order of presenting our experimental setup, we will go through the respective results. We first show the on-chip results for the reproduction of previous side-channel attacks using TDCs as well as the control condition with a PLL-filtered input signal, which acts as a comparison for the later results, followed by an extension to multiple boards and galvanically isolated HDMI communication. For SNR results and CPA-based attacks, we will always show the correlation result for byte 0 for comparability.

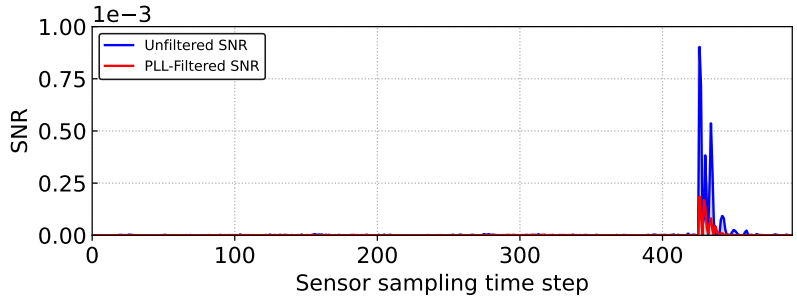


Figure 4.8.: Plot showing the unfiltered and PLL-filtered SNR of the On-Chip experiment

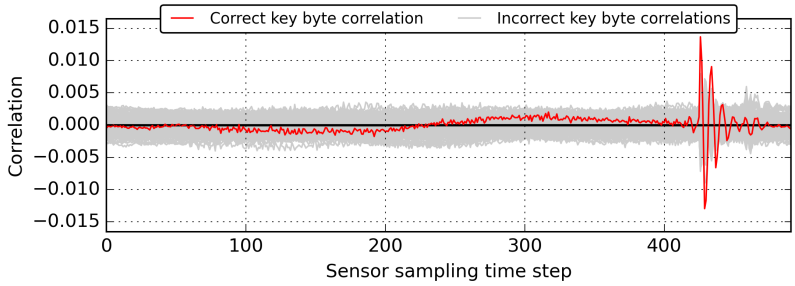
4.4.1 On-Chip Attacks

Figure 4.8 shows the SNR for the unfiltered condition and the condition using the PLL-filter for the on-chip condition. When comparing the SNR values, the PLL-filtered condition shows lower SNR. Thus, compared to the original clock, the PLL-filter seems to be less affected by voltage variations that contain side-channel information. However, the SNR is still considerably higher during the last round of AES as compared to the rest of the time. This means that both signals definitely carry leakage, considering that we are performing a known-ciphertext attack.

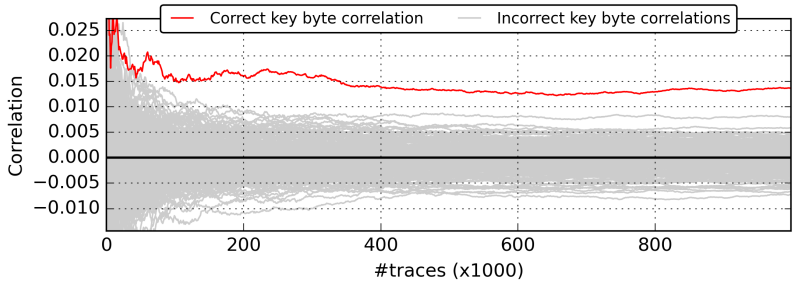
Figures 4.9 and 4.10 show the results for the CPA attack for the on-chip condition. Figures 4.9(a) and 4.10(a) show the correlation between the byte hypothesis and all 1 million collected traces for every timestep. Wrong byte hypotheses are displayed in gray, whereas the correct byte hypothesis is represented in red. This kind of plots will be called *CPA total-plots* from here, as they show the correlation for every timestep over the total of traces.

Figures 4.9(b) and 4.10(b) relate the correlation of a certain byte hypothesis with the number of traces employed in the analysis. These progress plots always refer to a single time step, so they can be thought of as a depth-slice of the corresponding total plot at that time step. The time step was chosen to be the one for which the correlation of the correct byte was at a maximum and located at the end of the trace, such that it must be during the last round of AES. If the total plot showed no points of interest for one of the delay lines, the same time step was chosen for the progress plot of both delay lines. Again, the wrong hypotheses are shown in gray and the correct hypothesis is shown in red. We will be calling this kind of plot *CPA progress-plot*, as they show the progress of the correlation coefficient over all the collected traces.

Figure 4.9 shows the CPA results for the delay line measuring the unfiltered clock signal. This is essentially the reproduction of previous power side-channel attacks and serves as a control for the experiments that follow. The figure clearly shows that the correct key byte can be recovered easily, as the red line stands out strongly in both plots. Not all bytes could be recovered during our experiments, however if any byte could be recovered, byte 0 usually showed the strongest leakage, which must be due to the specific AES



(a) CPA total-plot for the unfiltered delay line



(b) CPA progress-plot for the unfiltered delay line; the correct key byte leaks after 27k traces

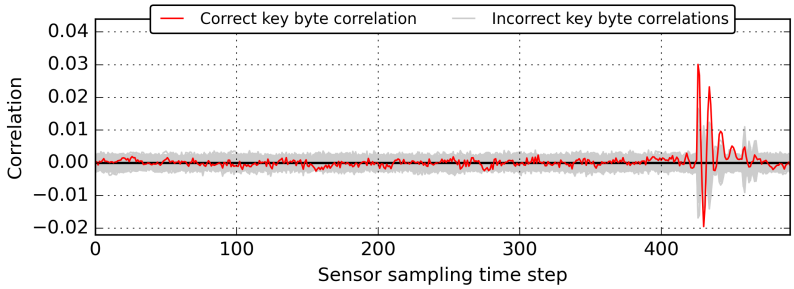
Figure 4.9.: CPA plots for over 1 million traces for the delay lines measuring the unfiltered signal for the on-chip experiment on ULX3S. The red lines represent the correct key guesses, while the gray lines represent the incorrect key guesses.

implementation. High absolute values of the red line around timestep 420 on the total-plot indicate the computation of the S-Box during the last round of AES. As the red line separates from the rest of the lines after about 40k traces in the progress plot, we can say that the attack is successful after 27k traces. This quantity is also reflected in Table 4.2.

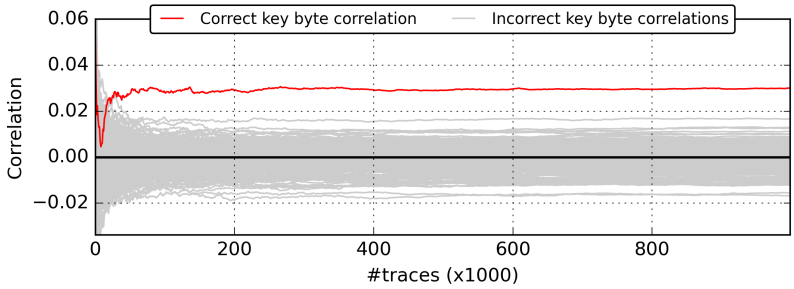
The results in Figure 4.10 from the on-chip separation condition are very similar to those from the on-chip baseline condition. In our experiment, the correct key byte for byte 0 could only be recovered after 40k traces. This difference to the previous experiment is not significant enough for a conclusion and is probably due to random effects. This result is expected, since the PLL we use for filtering resides in the same power domain and is thus again affected by the running AES module inside the chip, adding no benefit in terms of side-channel security. However, in the following results, this PLL-filtering serves as a valuable control experiment.

4.4.2 Wire-connected Attacks through a Twisted Pair Cable

In order to be more general, the results for the wire-based experiments have been collected on three different setups. In the first setup, victim and attacker are both ULX3S boards. In the second setup, the attacker is an ULX3S while the victim is a PYNQ-Z1 and for the last



(a) CPA total-plot for the PLL-filtered delay line



(b) CPA progress-plot for the PLL-filtered delay line; the correct key byte leaks after 40k traces

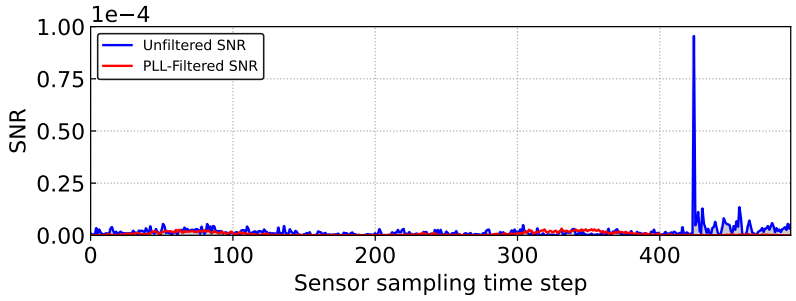
Figure 4.10.: CPA plots over 1 million traces for the delay lines measuring the PLL-filtered signal for the on-chip experiment on ULX3S. The red lines represent the correct key guesses, while the gray lines represent the incorrect key guesses.

setup, the roles are swapped such that the PYNQ-Z1 is the attacker and the ULX3S is the victim board.

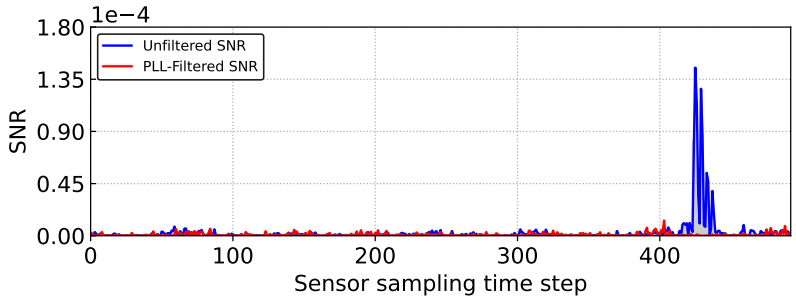
The SNR plots for the wire-connected attacks can be seen in Figure 4.11. The plots show a much stronger SNR for the unfiltered case during the last round of AES for all conditions compared to the PLL-filtered case. After evaluating the SNR results, we will now look at the CPA results.

First, we will look at the condition with the unfiltered input signal. The total-plots for the delay lines measuring the unfiltered signal Figure 4.12(a), 4.12(c) and 4.12(e) show relatively large correlation values during the last round of AES. The corresponding progress-plots Figure 4.12(b), 4.12(d) and 4.12(f) show the correct key byte being recovered after 318k traces for the experiment using two ULX3S boards and 317k traces for the experiment where the ULX3S is the attacker and the PYNQ-Z1 is the victim. For the last experiments where the PYNQ-Z1 is the attacker and the ULX3S is the victim, the key can already be recovered after 165k traces, which seems that the PYNQ-Z1 might have more side-channel leakage but also be able to observe leakage better.

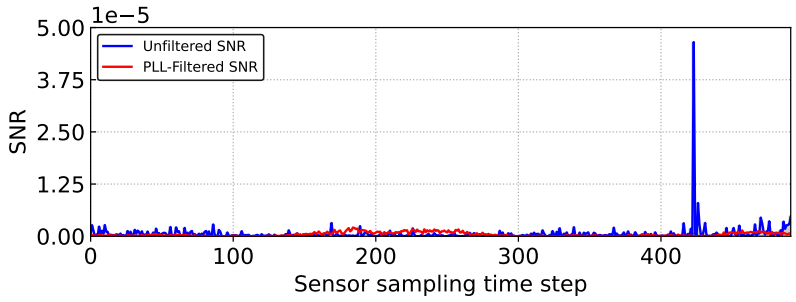
In contrast to the unfiltered signal, the total-plots for the delay line measuring the PLL-filtered signal in Figures 4.13(a), 4.13(c) and 4.13(e) do not show clear signs of correlation. The red lines in the progress-plots in Figures 4.13(b), 4.13(d) and 4.13(f) do not show any tendency and the correlation between the correct key byte power hypothesis and the



(a) SNR plot for attacker ULX3S #1 and victim ULX3S #2



(b) SNR plot for attacker PYNQ-Z1 and victim ULX3S #1

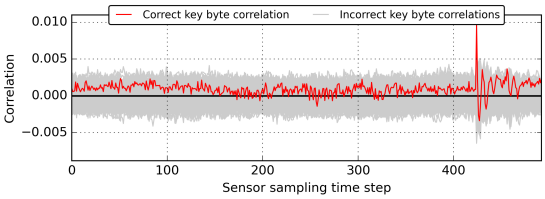


(c) SNR plot for attacker ULX3S #1 and victim PYNQ-Z1

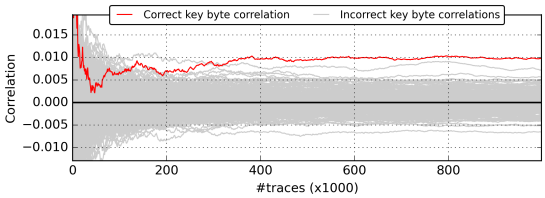
Figure 4.11.: SNR plots for the wire-connected experiments.

measurements at the given time step seem to converge towards 0. This is true for all key bytes, and not only the shown byte 0. These traces could not be used to reconstruct the key, even after collecting 1 million traces.

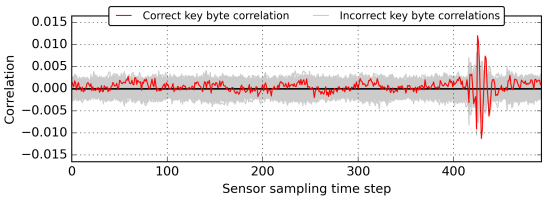
In order to investigate further, TVLAs were conducted by measuring 1M trace pairs, where 1M fixed messages and 1M random messages are alternately encrypted. Please note that typically more traces are recommended to show an absence of leakage, but this mainly acts as a crosscheck. The results for these can be found in Figure 4.14. The threshold in the total-plots for whether or not the signals are leaking is computed following a mini-p procedure. Traditionally, a significance level of 10^{-5} is required for the signal to be classified as leaky, so the threshold amounts to 5.61 for traces of length 494. For the progress-plots, the HC-procedure is employed. Here, the threshold is 334.



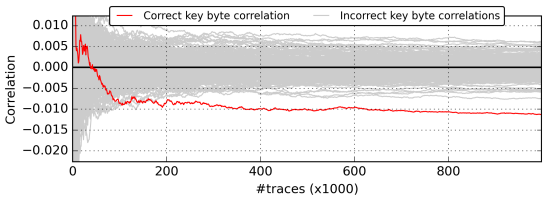
(a) CPA total-plot for attacker ULX3S #1 and victim ULX3S #2



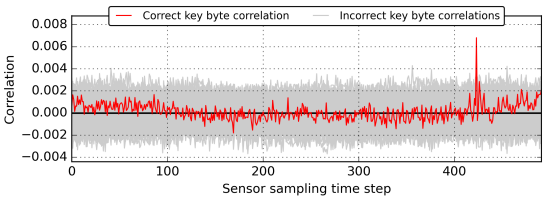
(b) CPA progress-plot for attacker ULX3S #1 and victim ULX3S #2; the correct key byte leaks after 318k traces



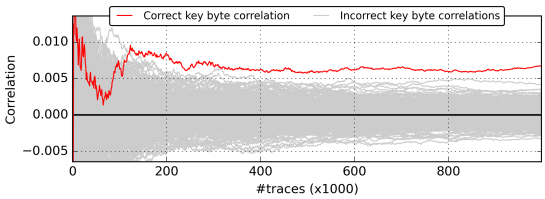
(c) CPA total-plot for attacker PYNQ-Z1 and victim ULX3S #1



(d) CPA progress-plot for attacker PYNQ-Z1 and victim ULX3S #1; the correct key byte leaks after 165k traces



(e) CPA total-plot for attacker ULX3S #1 and victim PYNQ-Z1



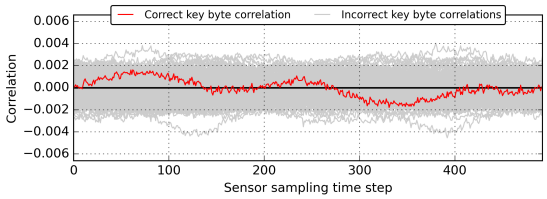
(f) CPA progress-plot for attacker ULX3S #1 and victim PYNQ-Z1; the correct key byte leaks after 317k traces

Figure 4.12.: CPA plots over 1 million traces for the delay lines measuring the unfiltered signal transmitted over single wires. The red lines represent the correct key guesses, while the gray lines represent the incorrect key guesses.

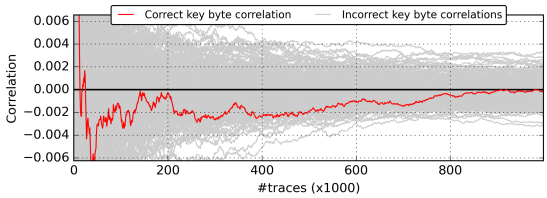
For the condition with two ULX3S boards shown in Figure 4.14(a) and its progress-plot in Figure 4.14(b), we notice values well above the significance levels of 10^{-5}, thus the leakage assessment is successful. The total-plot Figure 4.14(a) even shows some of the structure of the victim's computation. Before and after the encryption process, the victim does idle waiting. This can be seen between time steps 0 and about 80 and at the end of the total-plot. The reason for this assessed leakage could well be the fact that the PLL is no perfect jitter filter; furthermore, there is still a ground connection between attacker and victim, even though they do not share a power supply.

4.4.3 HDMI and isolated HDMI Attacks

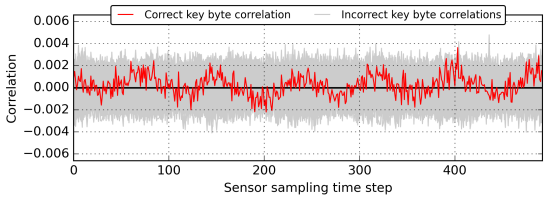
Finally, we look at results for the HDMI setup. Here, the PYNQ-Z1 is used as an attacker and the ULX3S #1 is the victim. We will discuss the results of the attack over a standard



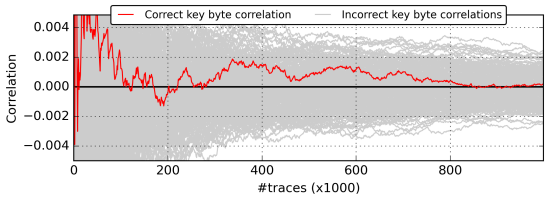
(a) CPA total-plot for attacker ULX3S #1 and victim ULX3S #2



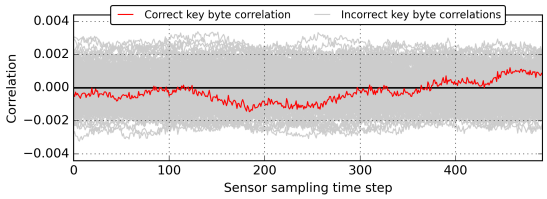
(b) CPA progress-plot for attacker ULX3S #1 and victim ULX3S #2



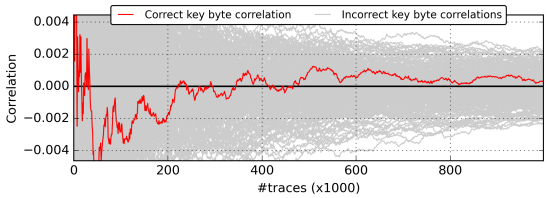
(c) CPA total-plot for attacker PYNQ-Z1 and victim ULX3S #1



(d) CPA progress-plot for attacker PYNQ-Z1 and victim ULX3S #1



(e) CPA total-plot for attacker ULX3S #1 and victim PYNQ-Z1



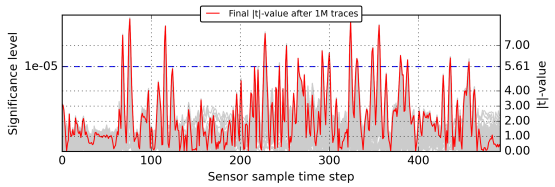
(f) CPA progress-plot for attacker ULX3S #1 and victim PYNQ-Z1

Figure 4.13.: CPA plots over 1 million traces for the delay lines measuring the PLL-filtered signal transmitted over single signal/GND twisted pair wires. The red lines represent the correct key guesses, while the gray lines represent the incorrect key guesses.

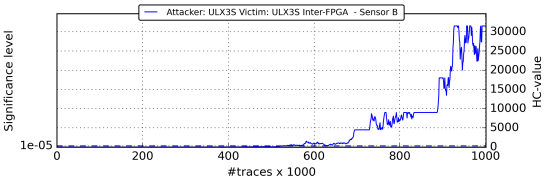
HDMI cable and the attack over the galvanic HDMI isolation module EVAL-CN0422 in tandem, as both are similarly successful. In addition to the EVAL-CN0422 for galvanic isolation, the victim board as well as the isolation board are placed inside an EM-isolated thermal chamber. Attacker, isolation board and victim are each powered by their own battery in order to eliminate any possibility of interaction through power lines.

For the HDMI based attacks, the SNR plots can be seen in Figure 4.15. Similar to the wire-connected experiments, the plots show a much stronger SNR for the unfiltered case during the last round of AES for all conditions compared to the PLL-filtered case. The SNR plots support our hypothesis that there exists leakage for the unfiltered case, while there seems to be no (or much less) leakage for the PLL-filtered case. After evaluating the SNR results, we will now look at the CPA results.

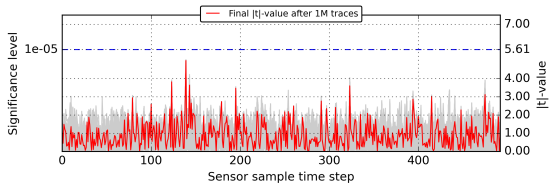
Both the results on direct HDMI in Figure 4.16(a) and galvanically isolated HDMI in Figure 4.16(c) show relatively large correlation values during the last round of AES, even more than those observed for twisted pair wires in Section 4.4.2, which might be due to an increased signal quality through differential signalling in HDMI. The corresponding



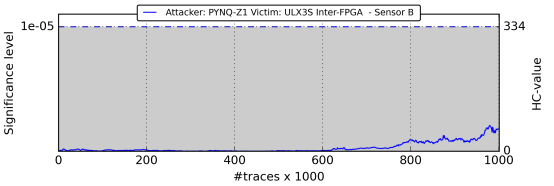
(a) TVLA total-plot for attacker ULX3S #1 and victim ULX3S #2



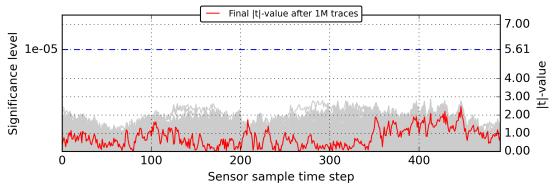
(b) TVLA progress-plot for attacker ULX3S #1 and victim ULX3S #2, leakage after about 550k



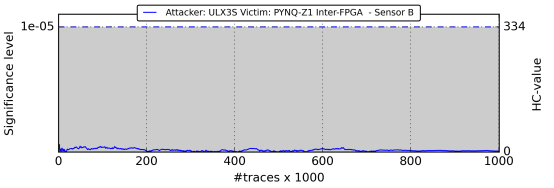
(c) TVLA total-plot for attacker PYNQ-Z1 and victim ULX3S #1



(d) TVLA progress-plot for attacker PYNQ-Z1 and victim ULX3S #1, no leakage can be reported



(e) TVLA total-plot for attacker ULX3S #1 and victim PYNQ-Z1

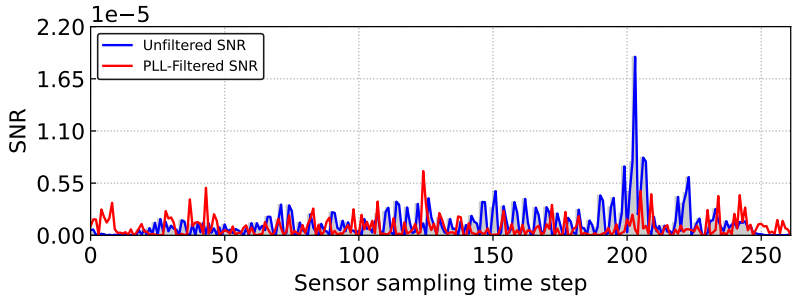


(f) TVLA progress-plot for attacker ULX3S #1 and victim PYNQ-Z1, no leakage can be reported

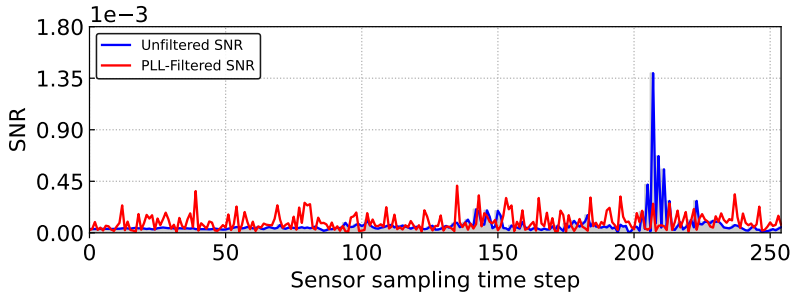
Figure 4.14.: TVLA plots over 1 million traces pairs fixed/random for the delay lines measuring the PLL-filtered signal transmitted over single signal/GND twisted pair wires.

progress-plots in Figure 4.16(b) and Figure 4.16(d) clearly show the correct key byte in red, separating from the gray lines of incorrect keys, indicating a successful recovery of the 0th byte of the key. Even though our effective sampling rate is only half of the previous experiments (50 MHz vs 100 MHz), the correct key can be recovered after about 48k in both cases. Thus, the galvanic isolation chip does not seem to reduce the jitter side-channel leakage in the original signal at all.

Despite we receive a NRZ-signal, we also put this signal through a PLL, which should work for clock-like phases (but not during communication phases). Here, in contrast, the total-plots for the PLL-filtered delay line in Figure 4.17(a) and Figure 4.17(c) do not show any signs of correlation. Also, the red lines of the correct key bytes in the progress-plots in Figure 4.17(b) and Figure 4.17(d) do not display any tendency and the correlation between the correct key byte power hypothesis and the measurements at the expected time step of the last AES round seem to converge towards 0. These traces could not be used to reconstruct the key, even after collecting 1 million traces. This confirms results from Section 4.4.2 that PLL-filtering could effectively reduce the chance for a successful CPA if we are only analyzing a clock signal, but as TVLA has shown before, it cannot completely remove it. Furthermore, please note that the filter is implemented on the side of the



(a) SNR plot for the signal transmitted using a standard HDMI-cable



(b) SNR plot for the signal transmitted via HDMI using an EVAL-CN0422 for galvanic isolation; victim and isolation board are placed inside an EM-isolated thermal chamber with only the HDMI cable exiting the chamber

Figure 4.15.: SNR plots for the experiments connected via HDMI.

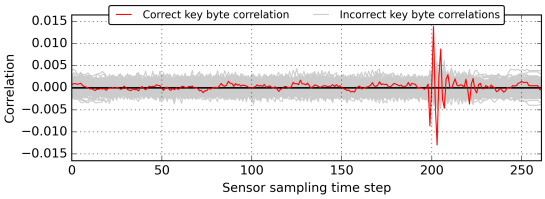
attacker, the victim itself cannot effectively use a PLL to filter within their own power domain, as we have shown in Section 4.4.1.

Since we already confirmed in Section 4.4.2 that the PLL cannot perfectly filter all side-channel leakage, we leave TVLA out in this experiment, since it would also add some difficulty to our experimental setup. These difficulties are the unidirectional communication on one hand, as well as the PLL that is connected to an actual data signal which would also contain fixed/random handshaking information that might keep the PLL in a state where it does not oscillate correctly for multiple clock cycles.

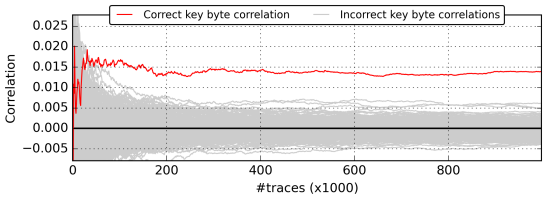
In total, this last experiment confirms our main hypothesis: Signal jitter in communication links leak side-channel information from the transmitter of this signal and can become a victim to this new class of side-channel attacks.

4.4.4 Summary of Results

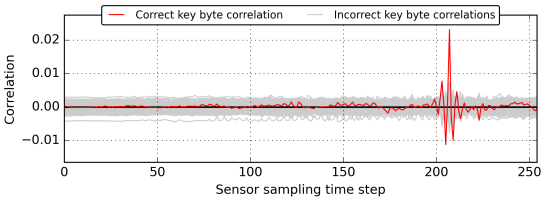
In Table 4.2 we summarize our results. This overview shows that on-chip attacks are equally successful whether jitter is filtered with a PLL or not, which is probably due to the PLL again being influenced by on-chip voltage fluctuations in the same way the original clock source is. For all the experiments that were cross device, filtering the jitter can reduce the chance of a successful attack.



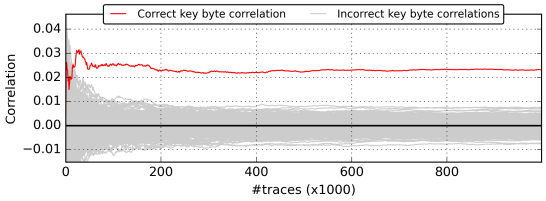
(a) CPA total-plot for the signal transmitted using a standard HDMI-cable,



(b) CPA progress-plot for the signal transmitted using a standard HDMI-cable; the correct key byte leaks after 47k traces



(c) CPA total-plot for the signal transmitted via HDMI using an EVAL-CN0422 for galvanic isolation; victim and isolation board are placed inside an EM-isolated thermal chamber with only the HDMI cable exiting the chamber

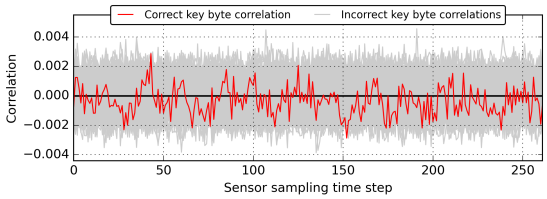


(d) CPA progress-plot for the signal transmitted via HDMI using an EVAL-CN0422 for galvanic isolation; victim and isolation board are placed inside an EM-isolated thermal chamber with only the HDMI cable exiting the chamber; the correct key byte leaks after 47k traces

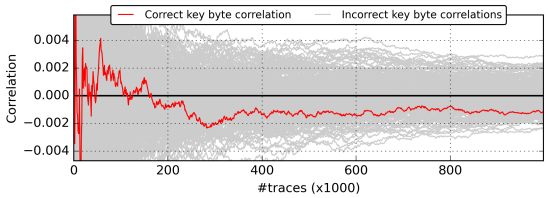
Figure 4.16.: CPA plots over 1 million traces for the delay lines measuring the unfiltered signal transmitted over HDMI. The red lines represent the correct key guesses, while the gray lines represent the incorrect key guesses.

These results clearly confirm our original hypothesis. In fact the jitter in the signal is the reason for side-channel leakage across devices, while this cannot be prevented if a differential signal, or galvanically isolated signal is used. Since the differential HDMI link leads to faster key recovery than the single-ended twisted pair wire, we would even assume, that high quality data links will also transmit the jitter at higher precision.

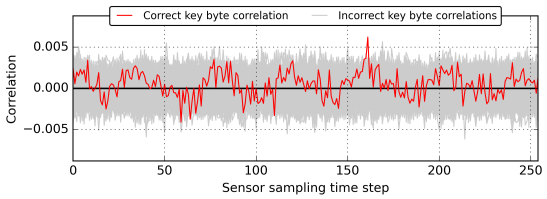
Table 4.3 shows descriptive statistics for the measured signal jitter values per experiment. We show the unit-less jitter from the sensors in which the mean is subtracted on a device-basis, as explained in Section 4.2.1. In all our experiments except for the on-chip condition the jitter's standard deviation decreases when filtering the signal using a PLL, indicating the voltage fluctuations having the most effect on the jitter. For the PYNQ-Z1 we can relate the results to a 11.5 ps estimate per bit, leading to a mean jitter of about 54±45 ps in the worst-case setup (isolated HDMI). Anyway, these statistical values of jitter cannot directly indicate whether a side-channel attack would be successful or not, when comparing the results in Table 4.3 to Table 4.2.



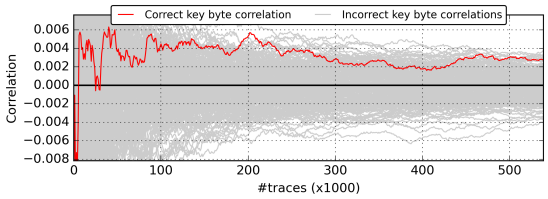
(a) CPA total-plot for the PLL-filtered signal transmitted using a standard HDMI-cable



(b) CPA progress-plot for the PLL-filtered signal transmitted using a standard HDMI-cable



(c) CPA total-plot for the PLL-filtered signal transmitted via HDMI using an EVAL-CN0422 for galvanic isolation; victim and isolation board are placed inside an EM-isolated thermal chamber with only the HDMI cable exiting the chamber



(d) CPA progress-plot for the PLL-filtered signal transmitted via HDMI using an EVAL-CN0422 for galvanic isolation; victim and isolation board are placed inside an EM-isolated thermal chamber with only the HDMI cable exiting the chamber

Figure 4.17.: CPA plots over 1 million traces for the delay lines measuring the PLL-filtered signal transmitted over HDMI. The red lines represent the correct key guesses, while the gray lines represent the incorrect key guesses.

Table 4.2.: Overview of our results for performing CPA with 1M traces and TVLA with 1M fixed/random trace pairs.

			Wire (Victim / Attacker)			**HDMI**	
		On-Chip	**ULX3S/ ULX3S**	**ULX3S/ PYNQ**	**PYNQ/ ULX3S**	**non-isolated**	**isolated**
CPA	unfiltered	~27k	~318k	~165k	~317k	~47k	~47k
	PLL-filtered	~40k	failed	failed	failed	failed	failed
TVLA	unfiltered	Yes	Yes	Yes	Yes	n/a	n/a
	PLL-filtered	Yes	No	No	No	n/a	n/a

4.5 Discussion and Countermeasures

Our results clearly show that galvanic isolation through a differential signaling link still allows side-channel leakage in the form of signal jitter being transported across these devices. In contrast to a previous work by Schmidt et al. [150] who performed a voltage

Table 4.3.: Overview of the jitter characteristics for all the experiments. All values are based on unit-less delay line sensor measurements. Jitter values are computed as the absolute difference between the sensor values and their mean for a given setup.

			Wire (Victim / Attacker)			**HDMI**	
		On-Chip	**ULX3S/ ULX3S**	**ULX3S/ PYNQ**	**PYNQ/ ULX3S**	**non-isolated**	**isolated**
unfiltered	min/max	0.36/6.64	0.11/16.11	0.15/16.15	0.42/23.58	0.41/26.59	0.38/27.38
	mean ± std.dev.	1.49±0.88	1.63±1.39	1.16±0.91	2.17±1.29	2.27±1.58	4.97±3.73
PLL-filtered	min/max	0.43/5.57	0.33/3.66	0.09/13.10	0.48/4.52	0.46/12.46	0.14/10.14
	mean ± std.dev.	1.86±1.31	1.56±0.62	0.51±0.45	1.46±0.75	0.63±0.39	0.27±0.43

measurement on a constant high signal that is connected through an optocoupler, we rather perform timing measurements on a signal that varies its level, i.e. clock or NRZ data, significantly lowering the attack barrier. We thus observe a new side channel, which derives the supposedly power side channel leakage from the signal jitter it causes in an outgoing signal.

We showed that direct-to-device communication is clearly at risk, which goes beyond HDMI and can affect other interfaces such as USB 3.0, Ethernet, and even interfaces within a device, such as PCIe or SDRAM. Any synchronized communication channel that contains jitter may also be at risk, potentially also fiber-channel interfaces. Furthermore, we believe that even circuit-switched networks might not entirely prevent this attack from being successful, even over multiple hops, due to the absence of buffering that would eliminate jitter through time discretization.

We also tried to separate the effects from jitter and power by using a PLL that has a certain level of jitter rejection as a filter of the jitter at its input. This is relatively successful in that CPA does not work anymore on the filtered signal. Like that, we can show that the component of power leakage that affects the delay line directly was prevented by separating the power domains. However, please note that this filter is only an experimental vehicle that cannot work as a countermeasure, since it can only filter clock signals. Furthermore, so far, it was applied on the attacker side, a practical design would need to add a third galvanically isolated domain that acts as an independently powered filter for the victim.

Since all of our experiments were performed using similar measurement designs and the same AES design, we can conclude that jitter leaks a very high degree of power side-channel information across boards. The required traces for successful key recovery are just about 2–10× more than those needed for FPGA-based on-chip power measurements, which is less of a difference than we expected. Interestingly, a higher quality connection

through a differential link on the HDMI cable seems to allow for easier attacks than a simple twisted pair wire connection. While it was not the focus of this work, future experiments might be able to show that a significant degree of side-channel leakage in the on-chip attacks benefits from clock jitter, and not just the impact of voltage fluctuations on the delay line itself, which is usually the assumption.

In summary, signal jitter can leak sufficient (power) side-channel information across boards, even without an actual power connection. By that, our work adds another new side-channel attack vector that was previously not addressed. Countermeasures that directly prevent this new side channel vector will need significant effort, as galvanic isolation cannot prevent it. If this attack vector gets developed further, millions of devices can be affected.

As a countermeasure, if actual messages and not a clock are transferred, we assume that a certain level of buffering and re-transmitting the message to be transmitted seems to be a feasible suggestion to prevent the side channel, which is hard to do for circuit-switched communication networks. The countermeasure may be using only packet-switched networks with at least one hop, like in most ethernet switches. After a hop, the jitter may be lost, which would prevent an adversary from carrying out a successful side-channel attack based on jitter. A very simple variant for such a countermeasure could also be adding a FIFO with depth 1 and two clocks with separated clock domains to decouple the jitter of the incoming signal from the outgoing signal. However, that FIFO would at least need to be on its own power domain, if not entirely EM-shielded.

Another aspect to discuss as a countermeasure is the use of data diodes [154], [155], [169], which secure a system by allowing information to flow in only one direction, and are already used in medical and military domains. However, here it should be noted that some implementations of data diodes might still allow jitter side-channel attacks. While any data diode would still enforce an information flow in only one direction, it might not prevent that more data (in the form of jitter) is sent in that same direction. That is essentially shown by our last experiments over unidirectional HDMI. However, we think that more complex data diodes which also buffer data [169], may effectively prevent jitter side-channels.

Overall, some of these isolation methods might work as a countermeasure, but further research is still required. Furthermore, it might still not be trivial to apply them in existing systems due to either practicability or cost issues.

4.6 Conclusion

Communication signals should not carry more information than what their specified bitrate allows. However, in this chapter we could show how minuscule timing variations of the rising and falling edges of a digital communication signal, also known as jitter, can carry sensitive information. First, we came up with a way to separate voltage- and jitter-based side-channel information. After that, we confirmed that we were able to

reconstruct a secret key used for encryption using the Advanced Encryption Standard by performing Correlation Power Analysis on the traces carrying only jitter-based side-channel information. Finally, we showed that this side-channel information is available even if the signal in question is fed through a galvanic isolator and both devices have no common potential reference. The results of this work solidify the assumption that galvanic isolation is no countermeasure for side-channel attacks. In addition, it points out that digital signals contain side-channel information that could potentially be sent over various high-speed signals such as optical links. This could become an additional security threat for millions of devices. Finally, it opens up a new medium for side-channel attacks that has not been explored before.

5 Automated Masking of FPGA-Mapped Designs

The work described in this chapter was first published in [5] and is joint work with co-authors Nicolai Müller, Dennis R. E. Gnad, Amir Moradi, and Mehdi B. Tahoori.

FPGAs play an important role in security-critical systems, as their reconfigurability enables security updates directly in the hardware and throughout the system's lifetime [170]. Hence, the overall system's security and flexibility are greatly increased by FPGAs. More concretely, if an FPGA implements a security-related operation, e.g. a cryptographic primitive operating on sensitive data, the circuit is updatable in case of a security flaw. However, an adversary, with physical access or remotely, can observe, disturb, or manipulate the FPGAs' physical characteristics to reveal sensitive data [171], [172]. For example, power analysis attacks exploit the data-dependent instantaneous power consumption of an FPGA [172]. Such an exemplary attack (which is one of many [171], [173], [174]) highlights the necessity for reliable protection to prevent physical attacks.

To prevent power analysis attacks, *masking* countermeasures are widely used [175]. Contrary to hiding, i.e. decreasing the SNR, formal security and adversary models abstract the masking's physical security, to formally evaluate and prove its correctness [176], [177]. Several hardware-based masking schemes – guaranteeing physical security under reliable adversary models – have been proposed and been applied to concrete algorithms such as the AES [178].

However, designing and proving an entire masked circuit by hand is costly, error-prone, and requires a high level of expertise in hardware security. Additionally, inefficient masked circuits, i.e. masked circuits requiring more resources than necessary, increase the overall costs for hardware and operating. To avoid implementation issues, it is desirable to build a large circuit out of masked and solely-proved sub-circuits. Unfortunately, composing provably secure sub-circuits does not necessarily guarantee that the resulting circuit is also secure [179]. Consequently, current research focuses on designing provably secure sub-circuits, so-called *gadgets*, which can be composed without violating the resulting circuit's security assumptions.

Usually, gadgets, e.g. *Hardware Private Circuits* (HPC1, HPC2 [180], and HPC3 [181]), realize basic logic functions which are provably secure and composable. Hence, gadget-based design libraries allow designers to transform an unprotected circuit into a masked circuit by replacing all gates with their equivalent gadgets. However, gadget-based masking imposes significant performance and power penalties. If done naively, the resulting

masked circuit needs a much higher number of clock cycles per execution compared to the unmasked design. For example, an AES substitution box (S-box) with cascaded two-input AND gates in four stages will result in an HPC-masked circuit running eight clock cycles (resp. four cycles) if the whole S-box is made out of HPC2 (resp. HPC3) gadgets. At the cost of area, GHPC [75] enables the transformation of an arbitrary function into a secure composable gadget with a constant amount of two register stages (resp. one for its clock-cycle reduced variant GHPCLL).

To apply composable gadgets in arbitrary designs, the open-source tool AGEMA[1] has been recently published to automate the generation of masked hardware dedicated to ASICs [182]. AGEMA receives the gate-level netlist of an unprotected ASIC design and provides the masked version of the same circuit while the user can select the desired gadgets, i.e., HPC1/2/3, and GHPC. Therefore, AGEMA facilitates implementing masked circuits even for hardware designers with little security expertise and reduces the risk of vulnerable designs. However, applied to FPGAs naively, AGEMA produces less than ideal designs, requiring more resources than necessary. Especially, GHPC has been initially defined and examined for ASIC platforms, and its relatively large area overhead, e.g. for realizing one AES S-box, makes it hard to apply to FPGAs. Alternatively, decomposing the S-box into smaller functions will decrease the area overhead at the cost of a higher number of clock cycles per execution.

Contributions: We present AGEMA_FPGA the first tool for the automated generation of efficiently masked designs on FPGAs. In order to minimize the usage of FPGA-components, the method behind AGEMA_FPGAprovides a trade-off between area and the number of clock cycles. For that, we decompose the target function, e.g. the AES S-box, into smaller functions that efficiently utilize the FPGA LUTs and apply GHPC on each LUT individually. We point out that no other method targets FPGA designs while ASIC-oriented tools, e.g. AGEMA, lead to a bad performance on FPGAs. In particular, AGEMA either masks the entire S-box with GHPC or every atomic non-linear AND gate with HPC. In summary, our contributions are:

- An EDA design flow for efficient (compared to AGEMA) automated mapping of arbitrary FPGA netlists to a provably secure composable masking scheme.
- The results are (1) provably secure confirmed by theoretical reasoning and experimental evaluation on an actual FPGA, (2) smaller compared to a GHPC-protected version of the entire function, and (3) faster compared to an HPC1/2/3-protected version of the target function.
- To automate the design process, AGEMA_FPGAimplements the ability to handle FPGA-based netlists. More precisely, we extend AGEMA features to protect an entire circuit with GHPC by the FPGA-optimized application of GHPC. Hence, AGEMA_FPGAnaturally extends the classical EDA flow.

[1] Available at: `https://github.com/Chair-for-Security-Engineering/AGEMA`.

- Open-source contribution: AGEMA_FPGAis available through the original GitHub repository[1].

5.1 Methodology: Automated Masking for FPGA Designs

This section illustrates our method behind AGEMA_FPGA to generate more efficient masked FPGA designs. We visualize the workflow of AGEMA_FPGA in Figure 5.1.

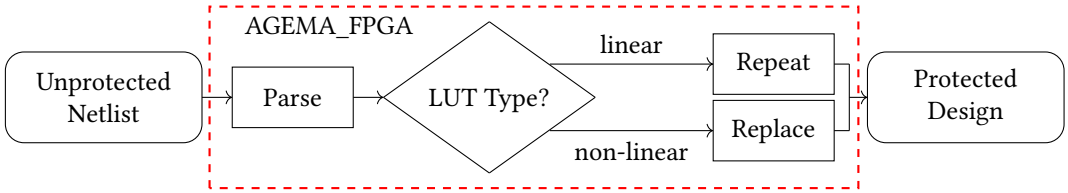


Figure 5.1.: Program flow of AGEMA_FPGA.

5.1.1 Parser

As stated, AGEMA deals with ASIC-based designs. More precisely, the netlist of an unprotected design, e.g. of AES, achieved through synthesis by Synopsys Design Compiler [DC] or Yosys [40], is given to AGEMA. When a behavioral-level design is synthesized by an FPGA synthesizer, e.g. Xilinx ISE or Vivado, the result (which can be either in Verilog or VHDL) is a netlist containing the FPGA-specific components including but not limited to:

1. LUTs with different sizes, e.g. LUT2, .. LUT6, LUT6_2.
2. Register cells, e.g. D FF with Clock Enable and Synchronous Reset (FDRE), Asynchronous Clear (FDCE), Synchronous Set (FDSE), Asynchronous Preset (FDPE).
3. Different multiplexers, e.g. MUXF7 and MUXF8, and
4. Buffer elements, e.g. BUFIO, INBUF, OBUF, etc.

However, the design cannot make use of additional specialized building blocks such as fast carry chains indicated by, e.g., CARRY4 or CARRY8 macros. First, we modified the parser of AGEMA to be able to read and understand FPGA-specific netlists. Such netlists can be generated at different steps. For example, Xilinx ISE can generate the netlists after Synthesis, Translate, Map, and Place-and-Route while Vivado does the same after Synthesis and Place-and-Route. Since original AGEMA deals with Verilog netlists, we also support only post-synthesis Verilog netlists (or other Verilog netlists generated at further design steps). Hence, the new parser reads the given Verilog netlist of an FPGA design and constructs a graph of the aforementioned FPGA-specific components and how they are connected.

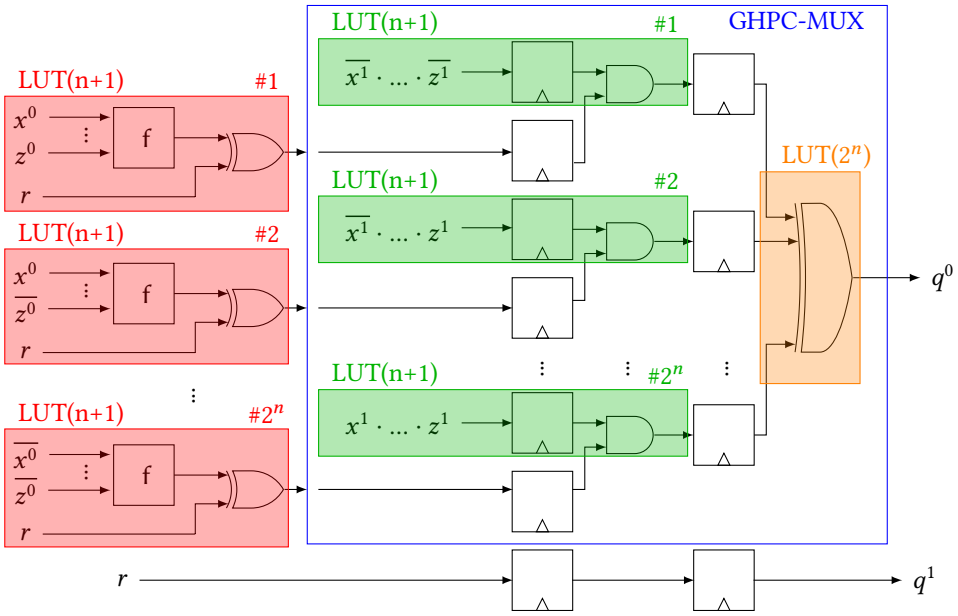


Figure 5.2.: Generic GHPC hardware module, partially taken from [75].

5.1.2 GHPC for FPGA constructs

As given in Section 2.4.3, GHPC is applicable on any arbitrary Boolean function f. Since every LUT realizes a Boolean function, GHPC can be applied on every LUT individually. More precisely, every LUT is replaced by a module shown in Figure 5.2, where every instance of the function f is an individual copy of the same LUT. As shown by the figure, different instances of f receive distinct set of inputs. For example, some receive x^0 and others its inverse $\overline{x^0}$ (the same for the other inputs). However, we can integrate such inversions into the LUT specification realizing the corresponding f. In other words, every LUT can receive the same set of inputs $x^0, \ldots, z^0$, realize the necessary inversions followed by f.

Figure 5.2 also shows that the output of every instance of f should be XORed by the fresh mask r. If f is a LUT with at most 5 inputs (LUT5), the entire inversion of inputs $x^0, \ldots, z^0$, the function f, and the corresponding XOR can be packed into a single LUT (in this case a LUT6), as indicated by red borders in Figure 5.2. Otherwise, if f is a LUT6, every red block should be implemented by two cascaded LUTs. Therefore, it may lead to more area-efficient results, if the given unprotected netlist does not contain any LUT6. Note that an LUT6_2 realizes a function $\mathsf{F} : \mathbb{F}_2^5 \rightarrow \mathbb{F}_2^2$, i.e., 2 Boolean functions with 5 common inputs. In such a case, every red block can be made by two parallel LUT6 instances, each of which is associated to a Boolean function. This would be the same if the aforementioned function F is realized by two distinct LUT5 instances. However, in case of the LUT6_2, the other part of the circuit marked by green borders in Figure 5.2 is reused for both Boolean functions. Hence, we avoid the usage of only LUT6 instances to achieve more efficient results. However, restricting the mapping to at most LUT5 is not needed for functionality.

Considering the blocks in green frames in Figure 5.2, the AND operation between $x^1, \ldots, z^1$ or their inversion can be implemented each by at most one LUT5, as we avoid the utilization of LUT6 in the unprotected design, i.e., Boolean functions f with at most 5 inputs. However, the register which stores the AND result, can be placed on the inputs $x^1, \ldots, z^1$. This way, we can integrate the single AND gate (at the right side of the green-bordered modules) into the LUT which makes for examples $\overline{x^1} \cdot \ldots \cdot \overline{z^1}$. We remark that for every Boolean function f with n inputs, 2^n instances of such a green module are required. This trick trivially saves $2^n - n$ registers (to save the results of e.g. $\overline{x^1} \cdot \ldots \cdot \overline{z^1}$) and 2^n LUTs (realizing the single AND gate as the output of such green modules). Finally, the 2^n-bit XOR gate in the orange frame is implemented by a set of cascaded LUTs.

5.1.3 Linear Functions

Cryptographic primitives, e.g., block ciphers, contain both non-linear and linear Boolean operations. As given in Section 2.4.2, when the non-linear gadgets are PINI secure, linear operations (XORs) can operate on each share individually. For example, a linear Boolean function $\mathsf{f} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ which should be applied on n-bit value $x, \ldots, z$, can be realized first-order secure using two shares as $q^0 = \mathsf{f}(x^0, \ldots, z^0)$ and $q^1 = \mathsf{f}(x^1, \ldots, z^1)$, since $q^0 \oplus q^1 = \mathsf{f}(x, \ldots, z)$.

In FPGA designs, such linear functions are also implemented by LUTs, which we refer to as **linear LUTs**. Trivially, one can apply the above-explained GHPC procedure to all LUTs including the linear ones. However, this leads to a high area overhead, a high number of clock cycles, and a high demand for randomness. Alternatively, we repeat the linear LUTs for each share leading to a significant area, clock cycles, and randomness reduction. Hence, in AGEMA_FPGA, the linear LUTs are detected, and instead of applying GHPC, they are instantiated two times (on each share independently).

To explain the detection of linear LUTs, we highlight that a LUT with n-bit input is implemented as a 2^n-bit memory followed by a 2^n-bit multiplexer, whose n-bit select signal is directly controlled by the LUT's input. The content of that 2^n-bit memory defines the function realized by the LUT, which is known as initialization value, so-called `INIT`.

Lemma 1. *Assuming that the LUT realizes the Boolean function* $\mathsf{f} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ *over n-bit input* $x, y, \ldots, z$, *the LUT is linear iff* $\forall x, y, \ldots, z \;\; \mathsf{f}(x, y, \ldots, z) = \overline{\mathsf{f}(\overline{x}, y, \ldots, z)} = \overline{\mathsf{f}(x, \overline{y}, \ldots, z)} = \ldots = \overline{\mathsf{f}(x, y, \ldots, \overline{z})}$.

In other words, inverting every single input bit should invert the output. By denoting the `INIT` by $\langle b_{2^n-1}, \ldots, b_1, b_0 \rangle$, we use Algorithm 3 to detect if the LUT is linear.

If the function realized by the LUT is an affine, i.e., a linear function XORed by a constant, the above procedure still detects it as a linear LUT. In such a case, if all LUT's inputs are 0, its output is 1. This can be easily seen through the `INIT` with $b_0 = 1$. Therefore, in order to apply the masking on a linear LUT, the same LUT with the same $\texttt{INIT}^0 = \langle b_{2^n-1}, \ldots, b_1, b_0 \rangle$ is

Algorithm 3 isLinear

Require: INIT=$\langle b_{2^n-1}, \ldots, b_1, b_0 \rangle$
Ensure: r : true if the corresponding LUT is linear
  $r \leftarrow$ true
  for $0 \leq \forall i < 2^n, 0 \leq \forall j < n$ **do**
    if $b_i = b_{i \oplus 2^j}$ **then**
      $r \leftarrow$ false
    end if
  end for

instantiated on the first input shares $(x^0, y^0, \ldots, z^0)$. The second instance of the same LUT is placed on the second input shares $(x^1, y^1, \ldots, z^1)$ with $\texttt{INIT}^1 = \langle b_{2^n-1} \oplus b_0, \ldots, b_1 \oplus b_0, 0 \rangle$. This would naturally cover both cases if the underlying function of the LUT is linear or affine.

5.1.4 Multiplexers

As given in [182], if the control signal of a multiplexer is not masked, similar to the linear functions, the same multiplexer can be individually instantiated on each share domain. More precisely, if we denote a multiplexer as $\text{MUX}(s, x, y) = q$ with s being the select signal, if s is not masked (i.e., an output of the Finite-State Machine (FSM)), we can write $\text{MUX}(s, x^0, y^0) = q^0$ and $\text{MUX}(s, x^1, y^1) = q^1$. This is still in conformity with the PINI notion.

If the FPGA-based netlist involves dedicated MUX modules, e.g., MUXF7, which are controlled by non-masked signals, we can already apply the method described in [182]. However, if such multiplexers are realized by LUTs they are not trivially detectable based on the module type. Hence, we need to detect them as well before applying the same technique. Otherwise, applying GHPC on such LUTs would needlessly increase the overhead. Since LUTs can be larger than 2-input multiplexers as well as 2-input XORs, a combination of linear functions and multiplexers are naturally merged and realized by a single LUT. For the same of simplicity, without losing generality, we focus on an example while explaining the procedure. Suppose that the underlying Boolean function of a LUT5 receiving three data signals x, y, z and two select signals t, w is

$$\mathsf{f}(x, y, w, t, z) = \begin{cases} x \oplus y & (t, w) = 00 \\ x \oplus z & (t, w) = 01 \\ x \oplus y \oplus z \oplus 1 & (t, w) = 10 \\ y & (t, w) = 11 \end{cases}.$$

In this case, INIT=$\langle b_{2^n-1}, \ldots, b_1, b_0 \rangle$= 1100 0110 0101 0110 1100 1001 1010 0110[2]. If t and w are not masked, we can – similar to linear LUTs – instantiate one individual LUT on each share domain, since every sub function of f categorized by (t, w) is linear. In order to detect this through the INIT, we can make use of Algorithm 4, where bit(j, k) returns the value of j at bit k. We remark that AGEMA already identifies t and w as (unshared) control signals as only sensitive parts will be masked.

Algorithm 4 isMux

Require: INIT=$\langle b_{2^n-1}, \ldots, b_1, b_0 \rangle$, $u = 2$, $v = 3$
Ensure: r : true if the corresponding LUT is a MUX
  $r \leftarrow$ true
  for $\forall w \in \{0, 1\}, \forall t \in \{0, 1\}$ **do**
    INIT' $\leftarrow \langle \forall b_{2^5 > j \geq 0} \mid \text{bit}(j, u) = w, \text{bit}(j, v) = t \rangle$
    if isLinear(INIT') = false **then**
      $r \leftarrow$ false
    end if
  end for

In the above given example, all four cases of INIT' = 0110 0110 / 0101 1010 / 0110 1001 / 1100 1100 are examined by Algorithm 3 which are reported to be linear. Hence, one LUT is instantiated on the first share domain (x^0, y^0, w, t, z^0) with the original INIT^0 = 1100 0110 0101 0110 1100 1001 1010 0110, and the second LUT on (x^1, y^1, w, t, z^1) with INIT^1 = 1100 **1001** 0101 0110 1100 **0110** 1010 0110. Note that the only difference between the LUTs is the green parts, since INIT' = 0110 1001 is an affine leading to **1001 0110** as the INIT for the counterpart on the second share domain.

This way, AGEMA_FPGA replaces the fully non-linear LUTs by their GHPC counterpart, and the others – which are a combination of linear and multiplexers – are highly more efficiently turned into a masked version by re-instantiating them with a slightly different configuration, i.e., INIT. This already highlights the importance of the way the unprotected implementation is given to AGEMA_FPGA. In other words, it should be tried to write the behavioral-level description of the unprotected design to combine the linear and multiplexer-like LUT. It is better if the LUTs are not a combination of non-linear functions and multiplexers, controlled by non-masked signals. Otherwise, more GHPC LUTs need to be instantiated, which naturally increases the hardware overhead. The same has been reported in [182], but with respect to the number of non-linear 2-input cascaded gates.

5.2 Evaluations

To examine the advantages of the designs generated by AGEMA_FPGA, we concentrate on the case studies provided in [182]. This includes the full cipher implementations of

[2] The colors should ease the identification of correspondence of INIT bits to the equations of f.

AES [183], CRAFT [184], Midori [185], LED [186], and Skinny [187]. Similar to [182], for the AES we cover both round-based and byte-serial implementations while only round-based implementation of other ciphers are considered. AGEMA_FPGA, generates every considered masked design within a matter of seconds, i.e. with a runtime that can be considered negligible. This is due to the fact that AGEMA_FPGA achieves linear complexity in the number of LUTs in the unmasked design.

5.2.1 Performance Evaluation

For benchmarking, we took the original GHPC-based designs, available on GitHub[1], and synthesized them using a Xilinx Virtex-6 FPGA (XC6VLX760). For the fresh masks, i.e., those random values which must be updated every clock cycle, we applied an FPGA-optimized LFSR design [188], where the entire 31-bit LFSR is realized by three LUTs, one for the XOR feedback function and two as shift register LUT SRL16E. To avoid reusage of fresh masks, we instantiated an individual LFSR for every required fresh mask bit, which is randomly initialized after the FPGA power-up. Afterwards, we took the behavioral description of the corresponding unprotected designs (also available on GitHub[1]), synthesized them by Xilinx ISE, and extracted the post-synthesis netlists. These netlists are then given to AGEMA_FPGA and the resulting designs are synthesized using the aforementioned FPGA fabric.

We performed these procedures for both GHPC and its low-latency version GHPCLL. The baseline unprotected results as well as the post-place-and-route results are depicted in Table 5.1 and Table 5.2. Note that the results include the resources utilized by the aforementioned LFSRs. It can be seen that – compared to the original designs – our generated new designs utilize less registers and LUTs and can operate at slightly higher frequencies. Furthermore, the overall power consumption is significantly reduced. The reason is that AGEMA_FPGA decomposes the S-box in a way that its mapping to a LUT-based representation becomes efficient. In particular, the restriction to LUTs with at most 5 inputs leads to comparably small GHPC gadgets as they grow exponentially with the number of inputs and allows the masking of individual GHPC components without the need of cascaded LUTs (cf. Section 5.1.2). This improvement is more noticeable in designs with larger non-linear functions, i.e., S-boxes with more inputs, particularly for the AES. We should, however, highlight that in such cases the number of required clock cycles of the generated circuit becomes higher. This is due to the S-box size, i.e., with an 8-bit input. More precisely, under the approach of AGEMA, GHPC is applied on the entire S-box, while AGEMA_FPGA first realizes the S-box by LUTs, on which the GHPC is applied. Since every coordinate function of the AES S-box with 8-bit input is realized by 5 instances of LUT6 in two stages, the application of GHPC on each LUT would lead to 4 additional clock cycles (resp. 2 cycles in GHPCLL). Hence, we confirm that our method finds tradeoffs between the number of clock cycles and the number of resources, which are particularly efficient when mapped to FPGAs.

Table 5.1.: Performance figures of different case studies generated by AGEMA (original) and AGEMA_FPGA (new) (including LFSRs) for GHPC

GHPC		Runtime [cycle]	Freq [MHz]	Mask [bit]	Reg [#]	LUT [#]	Power [mW]
AES (serial)	original	681	80	8	5 778	31 296	1 894
[183]	new	1 135	110	40	4 646	11 631	852
	diff	+ 67%	+ 38%	+ 400%	- 20%	- 63%	- 55%
AES	original	33	61	160	104 850	606 646	34 051
[183]	new	55	67	800	82 760	219 336	14 642
	diff	+ 67%	+ 10%	+ 400%	- 22%	- 64%	- 57%
CRAFT	original	96	171	64	2 381	2 473	157
[184]	new	96	188	64	2 293	2 163	138
	diff	0%	+ 10%	0%	- 4%	- 13%	- 12%
Midori	original	51	155	64	2 384	2 911	214
[185]	new	51	172	64	2 380	2 389	205
	diff	0%	+ 11%	0%	- 1%	- 18%	- 4%
LED	original	99	144	64	2 383	2 380	243
[186]	new	99	155	64	2 159	2 202	208
	diff	0%	+ 8%	0%	- 10%	- 8%	- 14%
Skinny	original	99	179	64	2 378	2 764	164
[187]	new	99	180	64	2 271	1 592	84
	diff	0%	+ 1%	0%	- 5%	- 43%	- 49%
Average	diff	+ 23%	+ 13%	+ 134%	- 11%	- 35%	- 32%

5.2.2 Security Evaluation

As stated in Section 2.4.3, the application of GHPC on an arbitrary function does not only make it first-order secure, but also in conformity with PINI notion, i.e., composable. Hence, it is proved that any circuit composed of only such PINI sub-circuits is also first-order secure. We should highlight that AGEMA and AGEMA_FPGA deal only with PINI modules and XORs, which also maintain PINI. Hence, the resulting circuit is secure under glitch- and transition-extended probing model. For the sake of completeness, we also conducted experimental analyses to examine the security of our generated circuits in practice. We made use of the evaluation board SAKURA-X [189] with a Kintex-7 target FPGA and examined the first-order leakage of our Skinny GHPC design. Since all designs are generated in a similar fashion, examining other designs would lead to the same result. We just did not select AES for this experiment, since the round-based design does not fit into the FPGA of our measurement setup, and the serial design would require very long traces, i.e., 1135 clock cycles, which makes a thorough evaluation unwieldy.

Using a digital oscilloscope, we monitored the current passing through a 1 Ω shunt resistor placed on the voltage source (Vdd) path of the FPGA. The traces have been collected at the sampling rate of 500 MS/s while the design was operating by a stable clock at the frequency of 3 MHz. Such a low clock frequency is selected to avoid distortion in measurements and

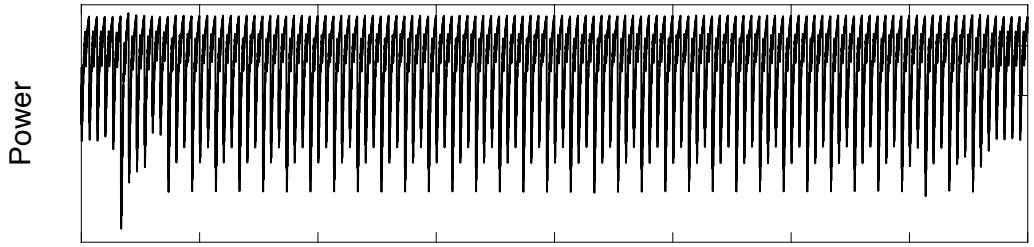
Table 5.2.: Performance figures of different case studies generated by AGEMA (original) and AGEMA_FPGA (new) (including LFSRs) for $GHPC_{LL}$

GHPC$_{LL}$		Runtime [cycle]	Freq [MHz]	Mask [bit]	Reg [#]	LUT [#]	Power [mW]
AES (serial) [183]	original	454	111	2 048	3 182	35 219	2 276
	new	681	118	2 560	2 597	15 158	1 184
	diff	+ 50%	+ 7%	+ 25%	- 19%	- 57%	- 48%
AES [183]	original	22	91	40 960	53 440	684 654	55 990
	new	33	93	51 200	41 799	290 160	22 956
	diff	+ 50%	+ 3%	+ 25%	- 22%	- 58%	- 59%
CRAFT [184]	original	64	180	1 024	1 291	5 040	320
	new	64	180	1 024	1 291	4 675	313
	diff	0%	0%	0%	0%	- 8%	- 2%
Midori [185]	original	34	171	1 024	1 294	5 744	375
	new	34	180	1 024	1 288	5 018	363
	diff	0%	+ 6%	0%	- 1%	- 13%	- 3%
LED [186]	original	66	121	1 024	1 293	5 791	498
	new	66	148	1 024	1 293	5 097	384
	diff	0%	+ 23%	0%	0%	- 12%	- 23%
Skinny [187]	original	66	173	1 024	1 416	5 027	368
	new	66	176	1 024	1 416	4 294	330
	diff	0%	+ 2%	0%	0%	- 15%	- 10%
Average	diff	+ 16%	+ 7%	+ 9%	- 7%	- 27%	- 24%

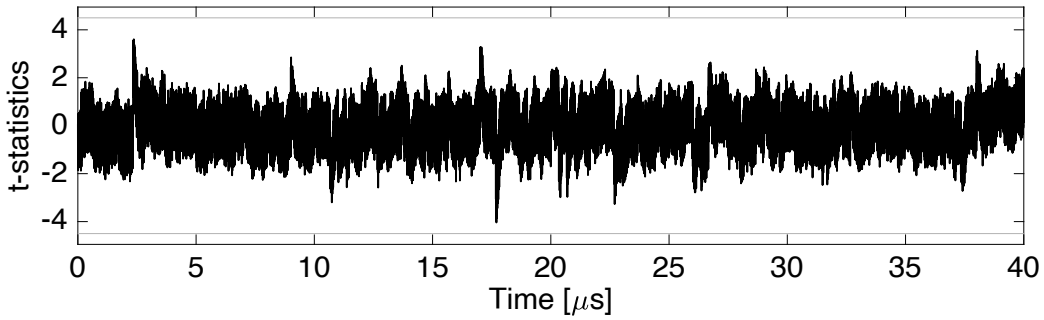
Table 5.3.: Unprotected designs performance as refference for different case studies in Table 5.1 and Table 5.2

Unprotected	Runtime [cycle]	Freq [MHz]	Mask [bit]	Reg [#]	LUT [#]	Power [mW]
AES (serial)[183]	227	296	-	276	430	23
AES [183]	11	258	-	284	1 347	122
CRAFT [184]	32	408	-	73	202	10
Midori [185]	17	289	-	76	413	31
LED [186]	33	253	-	75	336	32
Skinny [187]	33	525	-	134	172	8

collect clean traces (see a sample trace in Figure 5.3(a)). In order to examine the first-order security, we followed the instructions given in [59] and conducted a fixed-vs-random t-test (also known as TVLA) using 100 million power traces. Overall, this procedure took a couple of hours. In Figure 5.3(a) we can see the measurement of an average power trace, while in Figure 5.3(b), the leakage assessment over 100 million of such measurements show that no detection of leakage can be assessed, since $|t| = 4.5$ is not reached.



(a) Sample trace



(b) 1st-order t-test result, using 100 million traces.

Figure 5.3.: Experimental analysis on our Skinny GHPC design.

5.3 Conclusions

In this work, we presented AGEMA_FPGA, a novel tool for the automated transformation of an unprotected FPGA netlist into a masked design optimized for FPGAs. In particular, we guarantee the physical security of every generated design, verified in theory and on an FPGA test platform, under the PINI notion while optimizing for the efficient usage of FPGA components. Compared to the original outcomes of AGEMA, AGEMA_FPGA's results use up to 22% fewer registers, up to 64% fewer LUTs, and up to 59% less power while achieving comparable performance. Finally, we demonstrated the soundness of AGEMA_FPGA, which is already publicly available[1], by successfully evaluating a generated masked design on a Kintex-7 FPGA.

6 Design and Implementation of a Physically Secure Open-Source FPGA and Toolchain

The work described in this chapter was first published in [3] and is joint work with co-authors Daniel Lammers, Amir Moradi, and Mehdi B. Tahoori.

Protection against hardware attacks can be added in a dedicated part of an SoC or in the form of an embedded ASIC that handles sensitive information and executes cryptographic algorithms. However, such secure hardware is generally highly customized and tailored to a specific cryptographic algorithm or a limited set of functions. Hence, adapting to fast-developing security threats is difficult to impossible. While software updates are possible as a common reaction to new security threats, hardware usually remains unchangeable during its entire life cycle. As a result, secure hardware solutions can become outdated very quickly.

The hardware inflexibility issue can be solved with reconfigurable fabrics, e.g., FPGAs, that play an important role due to enabling rapid time-to-market by flexibility and updateability. FPGAs are often used for security applications and allow hardware "security patches" similar to software applications. Nevertheless, commercial FPGAs are not specialized for security applications and many security issues still need to be resolved [2], [27]–[31].

Integrating known countermeasures to physical attacks in conventional FPGAs is challenging and negatively affects performance significantly compared to solutions in ASICs, e.g., high area, low throughput, high power consumption, increased latency, etc. Implementation and mapping of such security schemes is usually re-done manually for every cryptographic algorithm and fabric architecture, which requires a high level of hardware security proficiency and development effort. In addition, developing designs dedicated to an FPGA platform is limited by the vendor tools and the primitives available on the device. Therefore, countermeasures might not provide the desired security level or lead to a high inefficiency with respect to performance.

For instance, there is a body of work focused on hiding countermeasures for conventional FPGAs, with some efforts exploring improvements through balanced placement and routing techniques [32]–[35], while others propose adjustments to the FPGA architecture or modifications to the implementation of LUTs [36]–[38]. However, these approaches are still not easily applicable to most commercially available FPGAs due to the aforementioned

limitations. Moreover, since they rely solely on hiding countermeasures, the provided side-channel resistance is not verifiable through a formal security model and can be considered as insufficient.

In this chapter, we propose a solution to address the challenges elaborated above, which serves as a roadmap to build a foundation enabling efficient realization of reconfigurable SCA-secure devices. The main aspects of our contribution are summarized as follows:

Secure Reconfigurable Fabric Design Methodology. We propose an approach to designing reconfigurable fabrics and a respective toolchain tailored with hardware security in mind. The concept provides resistance at the hardware level to various malicious physical attacks, such as SCA and FI attacks. A fabric designed by our methodology can act as the root-of-trust in SoCs in order to implement and execute cryptographic algorithms or functions demanding high security. Our technique preserves physical security at the gate level and is based on provable secure gadgets, subcircuits which are composable without violating security assumptions of the resulting circuit. Generally, many known gadget-based hardware security schemes can be utilized in the design process for such a fabric.

Fully Automated HDL Design Flow for Bitstream Generation. In fact, the mapping of a netlist to the fabric is crucial to maintaining security properties. If security assumptions are violated, physical security cannot be maintained. However, we made a customized toolchain to automate mapping as well as place-and-route for specific fabric architectures. As a result, no expertise is required in the hardware security domain, and insecure HDL designs are automatically mapped to achieve physically secure operation.

Entirely Open-Source Toolchain. We utilize AGEMA [24], FABulous [39], Yosys [40], nextpnr [41], and BitMan [42], [43]. These tools pave the way for efficient and customizable open-source development without vendor limitations. Our goal is to enhance security through comprehensive evaluation and expansion within the broader research community.

Secure Gadget-based Fabric Implementation. As a case study, we implemented a first-order SCA-secure reconfigurable fabric based on WDDL [44] and LMDPL [45] gadgets with the assistance of the above-mentioned open-source tools. The gadgets are adjusted to enable fault detection on the dual-rail signals according to the concept of Totally Self-Checking Circuits [46]. In this study, we present the complete design flow for secure fabric design. In particular, we explain the design of gadgets and how they are assembled into a reconfigurable fabric. Furthermore, we show the result of a fully automated bitstream generation flow for several cryptographic algorithms given their unprotected HDL designs. Moreover, we conduct an area improvement estimation. Compared to a conventional FPGA architecture (FABulous reference implementation), our mapped HDL designs result

in reduced area consumption by approximately 85%. TVLA on real power measurements of an emulated FPGA platform confirms no leakage detection over 100 million traces.

6.1 Preliminaries

6.1.1 Notation

In the context of this chapter, lowercase letters such as x denote single-bit (binary) variables in $\mathbb{F}_2$. The subscript indicates the rail of a variable in dual-rail form. The original value is given by x_t, while x_f corresponds to its complement. In condensed format, the two values are written as the tuple $x = (x_\mathrm{t}, x_\mathrm{f})$. In case of a shared variable, the index is explicitly stated in the superscript, e.g., x^0. Capital letters, such as X, represent a quadruple of variables split in two shares in dual-rail form each, i.e., $X = (x_\mathrm{t}^0, x_\mathrm{f}^0, x_\mathrm{t}^1, x_\mathrm{f}^1)$. The $\oplus$ symbol is utilized for additions in $\mathbb{F}_2$.

6.1.2 FABulous

FABulous [39] is a framework designed for eFPGA development. It integrates other well-known open-source tools such as Yosys [40], ABC [190], VPR [191] and nextpnr [41]. By this, it is capable of generating the eFPGA fabric for chip fabrication, matching bitstream generation and testing. The framework organizes its eFPGA fabrics into a grid of tiles. Each tile type is described via three components: (1) primitives, (2) wires, and (3) switch matrices. Primitives are combinational as well as sequential modules that implement the main functionality of the tile, and each tile may consist of one or multiple primitives. Defined wires are oriented in specified directions and, together with switch matrices, are used for configurable tile interconnections, while some wires can also be hard-wired.

The fabric designer implements different customized tile types and organizes them into the layout of a reconfigurable fabric. Usually, the same kind of tiles are placed row- or column-wise. Interconnections to adjacent tiles and global components are also specified for each tile type.

For end-users, FABulous supports the open-source F4PGA toolchain[1], which utilizes Yosys [40] and nextpnr [41]. This way, mapping of end-user Register Transfer Level (RTL) designs, as well as place-and-route, are fully automated. Eventually, the compatible bitstream generation is performed via BitMan [43].

[1] https://f4pga.org/

6.2 Methodology: Secure Fabric Design

In this section, we propose a general design process for reconfigurable fabrics tailored to the prevention of physical attacks. We provide resistance against individual SCA and FI attacks. The underlying security concepts are integrated into the design process, supported by open-source tools. Further, the fully automated flow of mapping an unprotected HDL netlist to secure fabric primitives is explained. In this work, we confine the scope to the primary goals, hence we assume a non-malicious bitstream. Details that are similar and in line with the original FABulous FPGA architecture and minor adjustments that give no further insights into the tailored approach are omitted.

6.2.1 Security Requirements

Our security requirements are derived from a comprehensive adversary model, which establishes assumptions about the potential attackers' knowledge, capabilities, resources, and limitations. By anticipating the methods that adversaries might use, we ensure that the system's defenses are designed to withstand a wide range of sophisticated attacks. The assumptions derived from this play a crucial role in shaping the security measures to meet the anticipated challenges. A detailed list of these adversarial assumptions, which form the foundation for the system's security requirements, is provided below:

Adversary Knowledge

- The adversary has knowledge about the general architecture of the reconfigurable device.
- The adversary has knowledge about the programmed bitstream configuration, including the cryptographic algorithms or computational processes it executes.

Adversary Capabilities

- The adversary has physical access to the target device and might depackage the chip to get access to its surface or is in close proximity to the chip.
- The adversary can observe the device inputs and outputs.
- The adversary is able to collect various types of SCA information, such as power consumption, electromagnetic emissions, timing information, and temperature, using appropriate sensors and measurement equipment.
- The adversary has the ability to process and analyze the collected SCA traces to infer sensitive information.
- The adversary possesses the necessary equipment to induce faults, such as voltage and clock manipulation tools, electromagnetic probes, and laser beams.
- The adversary can synchronize fault injections with specific moments during the device operation, based on timing information and observable behavior.

- The adversary can manipulate the device operating environment, e.g., temperature, power supply voltage, additional EM noise, etc.

Adversary Limitations

- Based on the defined security order d (i.e., the order of the applied masking scheme), the adversary is limited to place d probes on the internal signals of the device and monitor the corresponding values at specific clock cycles per encryption/decryption. In this work, we deal with first-order security, i.e., $d = 1$.
- The adversary cannot directly modify the internal state of the device (e.g., memory contents and register values) except through induced faults.
- The adversary does not have the ability to modify the hardware architecture of the device or its configuration, including the bitstream.
- The adversary can either observe SCA leakages or inject faults, but not both simultaneously.

As mentioned before, the system must be fully reconfigurable in the field, capable of implementing an arbitrary circuit while maintaining security properties. The only constraints are available resources, similar to a conventional FPGA. In this way, the following properties, which are essential for maintaining security, are preserved:

- Patches to algorithms implemented in secure hardware.
- Full exchangeability of security-related primitives, algorithms, protocols, etc.

Furthermore, there should be no need for the engineers/developers to have detailed knowledge about the security measures implemented in the fabric. The toolchain should be capable of automatically taking an unprotected target HDL design and map it to secure primitives, as well as handling placement and routing according to the security measures. This ensures robust protection against SCA and FI attacks at the hardware level. Notably, automating the application of security measures to the design in such a manner significantly reduces the risk of human error in the process.

6.2.2 Platform Description

The secure reconfigurable fabric is supposed to act as a root-of-trust in an SoC. It should provide protection during the execution of cryptographic algorithms or other functions demanding high security. Fundamentally, this fabric implements a combination of first-order masking and hiding countermeasures to provide strong protection against SCA attacks and enables detection of fault injections. Usually, not every part of an implementation needs SCA protection. Therefore, we consider multiple regions with different security levels (cf. Figure 6.1), namely full-secure, control-secure, and non-secure. This concept is similar to microprocessors with TrustZone technology [121]. In this way, the associated total overhead and cost of the fabric are reduced, since building blocks require more area with increasing security.

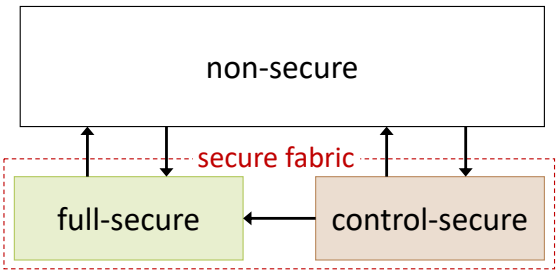
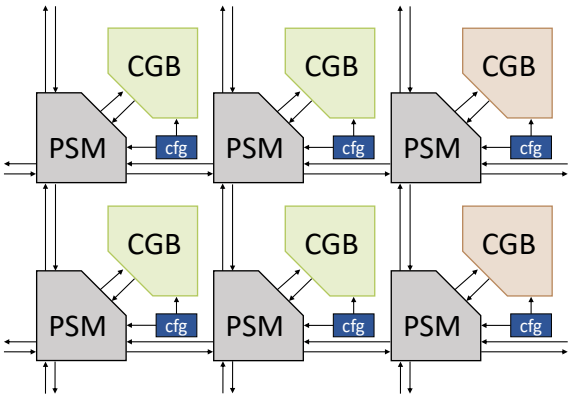
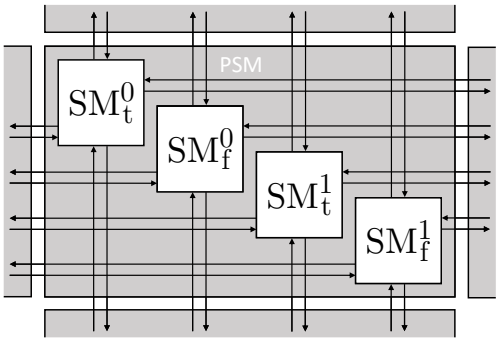


Figure 6.1.: Schematic of regions with different security level.



(a) Secure fabric organization of different Configurable Gadget Block (CGB) types, Parallel Switch Matrix (PSM) and corresponding configuration memory (cfg).



(b) Share-wise domain separation and dual-rail preservation in a Parallel Switch Matrix (PSM).

Figure 6.2.: FPGA-like grid structure of secure fabric.

If protection against physical attacks is not required for a part of the implementation, the non-secure region is the trivial choice. It consists of non-secure logic, such as any kind of processor core or conventional FPGA. The control-secure region is supported by a fault detection mechanism as a countermeasure against FI attacks. This is mandatory to protect parts of the secure design, such as the control logic of a cipher, where secret data are not processed, but faults may directly affect them. Therefore, we propose the application of DRP logic and fault propagation based on invalid DRP states in this region to support a wide range of fault detection. The full-secure region implements a combination of first-order secure masking and DRP. Fault occurrence is intended to propagate from the control- and full-secure regions to a fault detector located at the primary outputs. This will be extensively explained in Section 6.2.6.

In general, different approaches are possible with either less or more layers of security (including higher-order masking) to suit the desired/required protection level of an actual SoC. In this paper, we limit our focus to the full-secure and control-secure regions, which fulfill the purpose of an embedded secure reconfigurable fabric. We omit dealing with a non-secure region, as several solutions (like the original FABulous FPGA) can easily handle this. In short, we assume all inputs (and outputs) of our fabric follow a DRP protocol while some of them are additionally masked.

6.2.3 Secure Reconfigurability

To achieve reconfigurability of the fabric and maintain provable security at the same time, secure composable gadgets are used. The main idea, which is shown Figure 6.2(a), is to include each gadget in a Configurable Gadget Block (CGB) while the CGBs are organized in an FPGA-like grid structure and interconnected through configurable Parallel Switch Matrices (PSMs) (explained in detail in Section 6.2.4). CGB and PSM configuration data are stored in dedicated configuration memory (denoted as "cfg" in Figure 6.2(a)), which is programmed by the bitstream. Since gadgets are provably secure subcircuits that are composable under a specified composability notion without violating the security assumptions, any valid fabric configuration should maintain SCA-secure functionality.

For our purposes, we consider two general gadget types. The linear gadgets implement the XOR function and non-linear gadgets realize the AND functionality. Both of which require the inputs and the output to follow DRP logic. As a dual-rail inversion is implemented by swapping the rails, the gadget reconfiguration is realized by individually configurable multiplexers on the inputs and output. This would allow realizing XOR and XNOR with the linear gadget and AND, NAND, OR, and NOR with the non-linear gadget. The basic structure of a full-secure reconfigurable gadget is shown in Figure 6.3. The select signal of each multiplexer is set by a particular bit in the programmed bitstream. For the masked gadgets, only one share is invertible, since inverting both shares would result in the same orignal value and cause unnecessary overhead, i.e., $x = x^0 \oplus x^1 = \overline{x^0} \oplus \overline{x^1}$. Multiplexers at the reconfigurable gadget output rails control whether register stages are used. Specifically, two register stages are utilized to support alternating pre-charge/evaluation phases of DRP logic. Therefore, by composing these linear and non-linear gadgets followed by the aforementioned registers, any Boolean function and sequential circuit can be realized. More implementation details on a particular case study gadget designs are given in Section 6.3.2.

In general, a fabric CGB can consist of multiple gadgets and gadget types, by utilizing some local routing or fixed wiring. However, this is an application-specific optimization that does not affect security. Therefore, we consider only one gadget per CGB to limit the complexity.

6.2.4 Connectivity Constraints

To maintain provable security and functional correctness for any design, interconnections between the secure reconfigurable gadgets are restricted. The constraints are derived from composability notions and the gadget-based scheme used. For gadgets that rely on a combination of masking and DRP, this can be summarized as follows. Note that other schemes and composability notions may result in different connectivity constraints.

- *Share-wise domain separation.* A signal belonging to one share domain in one gadget should not be connected to another share domain of another gadget.

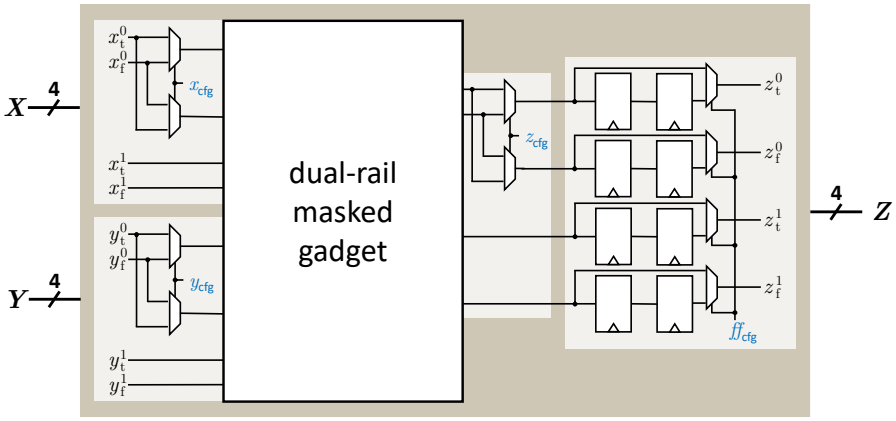


Figure 6.3.: Full-secure CGB with DRP logic and first-order masking. A central gadget implements linear or non-linear functionality. Multiplexers enable inversion of inputs and output. In addition, four multiplexers at the outputs control whether register stages are used. Configuration bits are highlighted in blue.

- *Dual-rail preservation.* It should be assured that dual-rail signals are interconnected in the correct way and no unintended rail swap occurs.

We propose to employ PSMs, illustrated in Figure 6.2(b), which are in line with the above constraint list. A PSM is a structure of parallel switch matrices in which only wires with the same share domain and the same rail indicator are interconnected, e.g, a configuration for $\mathrm{SM}_{\mathrm{t}}^{0}$ determines the true rail connections of the share domain 0. This results in several smaller switch matrices that are interconnected only with other switch matrices or CGB inputs and outputs, each forming an independent grid.

Lemma 2. *The routing of signals through distinct physical wires and dedicated switch matrices, corresponding to their respective share domain and rail, ensures complete physical separation. This assures that both share-domain-based separation and dual-rail preservation are maintained, making any violation physically impossible.*

Proof. Each gadget input or output, corresponding to the share domain $\sigma \in \{0, 1\}$ and the rail $\lambda \in \{\mathrm{t}, \mathrm{f}\}$, is connected exclusively to the wires x_{λ}^{σ} dedicated to that specific share domain σ and rail λ. These wires are further connected only to a switch matrix $\mathrm{SM}_{\lambda}^{\sigma}$ that also corresponds to the same share domain σ and rail λ. In particular, in the control-secure region, each dual-rail signal is carried by two physical wires, i.e., x_{t}^{0} is routed through $\mathrm{SM}_{\mathrm{t}}^{0}$ while x_{f}^{0} is routed through $\mathrm{SM}_{\mathrm{f}}^{0}$. In the full-secure region, the additional wires x_{t}^{1} and x_{f}^{1} are similarly routed through $\mathrm{SM}_{\mathrm{t}}^{1}$ and $\mathrm{SM}_{\mathrm{f}}^{1}$, respectively. In this way, it is physically impossible to violate share-domain-based separation and dual-rail preservation. □

Certain connectivity aspects demand particular attention, such as connections between different security regions. In particular, we consider a unidirectional connection from the control-secure to the full-secure region and no connection in the opposite direction. Processing secret data in the control-secure region is not allowed according to the platform description given in Section 6.2.2 and may cause SCA leakage.

6.2.5 Random Number Generator (RNG)

For masking to be applied, random numbers are required. In addition, many masking schemes additionally require fresh masks updated every clock cycle to maintain the claimed security. We consider randomness to be provided by a dedicated module outside the actual reconfigurable fabric. The True Random Number Generator (TRNG) alone seems to be a less efficient solution, as they are either quite slow or require extensive resources. In practice, this results in high costs for each true random bit generation. Since we need a large number of random bits, a combination of TRNG and scalable Pseudo-Random Number Generator (PRNG) is suitable for such a task. It is a common strategy to use more efficient PRNGs to stretch the initial truly random seed into many pseudo-random bits. It should be noted that the statistical quality of PRNGs (such as passing some National Institute of Standards and Technology (NIST) tests [192]) is important for masking schemes to keep their security promises.

In our proposed fabric design, externally provided randomness is expected to seed a scaled internal PRNG, which continuously generates fresh randomness during runtime and is directly hardwired to all non-linear gadgets. In addition, all inputs that enter the full-secure region from outside (i.e., primary inputs) should be provided in the masked representation.

6.2.6 Fault Propagation and Self-Checking

Dual-rail logics provide an opportunity for self-checking. Considering (x_t, x_f) as the dual-rail representation of x, we define a valid encoding based on the two valid states $(0, 1)$ and $(1, 0)$ as well as the two invalid states $(1, 1)$ and $(0, 0)$. By checking the correctness of dual-rail states in the circuit, faults are detected up to a certain extent. If any dual-rail state in the evaluation phase corresponds to an invalid encoding, a fault is detected.

Since examining the validity of every dual-rail is quite complex, the concept of Totally Self-Checking circuits [46] can be applied to ensure fault propagation. In this way, any invalid state should propagate through the circuit, making the output state of all successor gates also invalid. This would allow instantiating fault detector modules only at the primary outputs to evaluate the validity of the dual-rail states and hence detect faults. Note that neither WDDL nor LMDPL gadgets fulfill this requirement. In order to be in line with these principles, we adjust the secure gadgets to follow this fault propagation scheme. If no fault occurs, for every valid gadget input, we expect a valid gadget output. In case of an invalid input encoding, the output should also be invalid. In a composed circuit, this behavior causes fault propagation through the entire design. Once an invalid state propagates to the outputs and is detected, all embedded fabric IO ports can be disabled. It is important to note that the output evaluation requires an additional clock cycle. Consequently, the final outputs must be delayed by at least one clock cycle to ensure that the disable operation is executed before the outputs are revealed to the IO ports. Additionally, since gadgets represent atomic functions but are not actual atomic gates, any individual atomic gate within a gadget could potentially be faulted. Therefore, it is desirable to design the internal

circuit of the gadget such that any fault within either results in a faulty dual-rail state at the output or does not affect the output correctness. Otherwise, this would limit the fault detection capabilities. In total, any effective single fault within a gadget is propagated to the primary output of the composed circuit, due to the fault propagation property of each gadget.

However, a deployed fault detector module may violate the security assumptions and cause SCA leakage if share domain separation is not taken into account. In particular, the XOR operation applied to dual rails of any share always results in a logical '1' if a valid encoding is present during the evaluation phase. However, considering the glitch-extended probing model, a check on all outputs potentially leaks information through combining all shares of different domains.

It should be noted that such a technique cannot detect any faults injected on both rails of the same share simultaneously, e.g., changing a valid dual-rail signal $(0, 1)$ to another valid state $(1, 0)$. On the other hand, this is not always easily possible without affecting other signals. In other words, even if a fault injection causes a valid encoding to toggle to another valid encoding, it is enough to have a single invalid $(0, 0)$ or $(1, 1)$ signal in the entire circuit to detect the injected fault.

6.2.7 Toolchain and Design Mapping

Since in the proposed approach resistance against physical attacks is provided at the gate level, regardless of the design mapped to the fabric, its implementation provides the desired level of provable security during runtime. During the design mapping step for a provided secure fabric, the developer of the HDL design to be mapped does not require any specific knowledge about hardware security. Only the primary inputs have to be annotated as secret or public in the unprotected HDL design. Then, the tool automatically determines which parts of the given HDL are mapped to which security region based on these annotations. Figure 6.4 illustrates a flow of how mapping of a given HDL design to secure gadgets is done. As a first step, the HDL design needs to be mapped to specified atomic gates such as AND, NAND, OR, NOR, XOR, XNOR and NOT. This can easily be done with Yosys. In the next step, each of these gates is replaced by a reconfigurable gadget, explained in Section 6.2.3. This is realized by the instantiation of Verilog primitives according to the gadgets with the corresponding generic parameters in the netlist. The result is later automatically translated into the particular configuration in the further design process. Additionally, every wire is replaced by the corresponding number of wires, which connect gadgets in the same way as gates were connected before. This step can be supported by the open-source framework AGEMA, which we extended for our specific use case, as it is publicly available on GitHub.

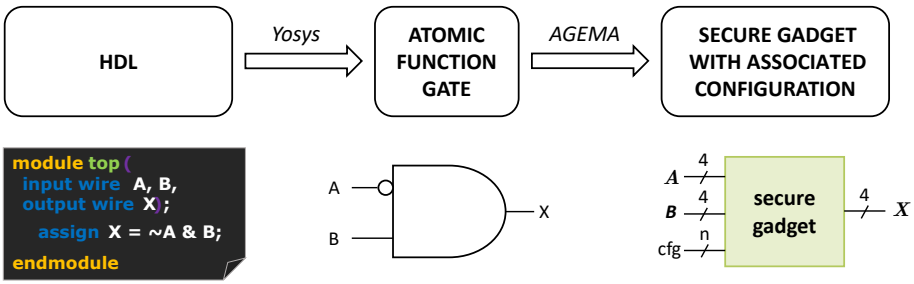


Figure 6.4.: Unprotected HDL design mapping to secure gadgets assisted by Yosys and AGEMA.

6.2.7.1 Place-and-Route

The result of the aforementioned steps is a netlist of the given design made solely by the primitives available in the CGB. As next, every primitive should find a place in the fabric and the PSMs should realize their interconnections. Similar to mapping, place-and-route is automated by open-source tools, for example, nextpnr or VTR in the FABulous ecosystem. However, in order to enhance SCA security with respect to the underlying DRP, we pursue a balanced routing without different routing delays for dual-rail signals. Even though our architecture enables the same routing for dual-rail signals, it is not assured that parallel routes will be taken by the routing algorithm.

An option to achieve parallel routing is to adopt the fat-wire approach [64]. We consider not only two, but also more associated wires of the same logic variable bundled together for routing and call it *bundle-routing*. First, every dual-rail is bundled and then treated as a single wire. Second, if a logic variable is masked, the dual-rail bundles for all shares are merged together (only as abstraction). Third, the *bundle-routing* routes each bundle as a single wire. Lastly, each bundle is resolved and the determined *bundle-routing* is translated to each contained wire to the actual corresponding physical parallel wire. This leads to parallel routes on the fabric for each rail and share according to the same logic variable.

At the physical level, there may still be some differences dependent on the actual physical implementation of switch matrices and manufacturing differences. Hence, balanced physical routes (which is relevant when fabricating the chip) should support our proposed bundle-routing technique. Note that – as stated – balanced routing in our approach is not a must as our SCA security relies on the provable secure composable gadgets. Hence, balanced routing may further harden higher-order SCA attacks to succeed in practice. Another problem is partial reconfiguration at run-time, which cannot be allowed, since the compossibility notions may be violated during the reconfiguration process. Therefore, we exclude this option to maintain security.

6.3 Case Study: Implementation of a Secure Fabric with Open-Source Tools

To evaluate our concept in practice, we implemented a secure reconfigurable fabric with the assistance of the FABulous framework. For this, we implemented custom gadgets and adapted the toolchain to follow our approach explained in Section 6.2. We map several designs on our custom fabric implementation and the original FABulous fabric as a reference to compare the area consumption in the special use case of physical security. Lastly, we emulate our custom FPGA on a real FPGA device, make use of our entire customized toolchain, and conduct an experimental SCA evaluation of the generated secure designs.

6.3.1 Design Decisions

First, we would like to highlight that we focus on area requirements and low-latency for the secure platform of our case study. Hence, our solution is based on the LMDPL [193] masking scheme that, independent of the circuit size, introduces only one additional cycle of latency for every cycle in the original design, ensuring minimal performance impact. The scheme also offers relatively small area overhead, which makes it an efficient choice for hardware implementations. However, similar to other gadget-based masking schemes, LMDPL requires fresh randomness (updated for every evaluation phase), namely 1 bit per non-linear gadget. For the (non-masked) control-secure region, we adopt WDDL [44] gates, as their area requirement is considerably lower compared to other DRP logics. Additionally, we modified both schemes (based on Section 6.2.6) to include fault propagation and detection mechanisms with minimal extra area cost, allowing to additionally resist against some FI attacks. As proposed in Section 6.2.5, the integrated PRNG is part of the reconfigurable fabric as a dedicated (non-reconfigurable) module. This reduces the overall fresh randomness generation overhead significantly compared to any solution that is implemented using reconfigurable primitives, such as LUTs. We rely on an unrolled Trivium stream cipher design, as suggested in [194]. We constructed the PRNG module also in DRP logic, which provides fresh randomness in every evaluation phase for every non-linear gadget and ensures having its outputs fully nullified during the pre-charge phase. Due to the fault propagation property of the gadgets, a faulty dual-rail state of the fresh randomness propagates (starting at every non-linear gadget input) through the gadget, ensuring detection at the primary outputs of the mapped design, similar to data input faults. For the sake of completeness, we provide a comparison between low-latency masking approaches applied on the AES S-Box in Table 6.1.

6.3.2 Gadget Design

Based on the principle of secure reconfigurability, explained in Section 6.2.3, we matched gadget-based masking to the grid-based approach in the FABulous framework, which

Table 6.1.: Comparison of first-order protected AES S-Box implementations using different low-latency masking schemes.

Reference	Scheme	Latency [cycles]	Area	Randomness [bits/cycle]	Fault Propagation
[193]	LMDPL	1	baseline	36	✗
[76]	LMDPL	2	-19%	36	✗
[195]	GLM	2	+94%	416	✗
this work	LMDPL_{SC}	1	+40%	36	✓

leads to the deployed CGBs. Each of our-designed CGB types contains one of the gadgets described below. As described in Section 6.2.2, we divide the platform into control-secure and full-secure regions. The control-secure region is based on Self-Checking Wave Dynamic Differential Logic (WDDL_{SC}) gadgets which operate on dual-rail signals with a fault-detection facility only. The full-secure region contains Self-Checking LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL_{SC}) gadgets that operate on masked dual-rail signals and offer first-order SCA security and fault detection. This design decision allows us to implement control logic that does not process secret data, e.g., a counter, based on the primary input annotation of the to-be-mapped RTL design with significantly less overhead. In the following, the exact gadget implementation and a deep analysis of their fault propagation feature are illustrated, while preserving the SCA security of the underlying scheme.

6.3.2.1 Self-Checking Wave Dynamic Differential Logic (WDDL_{SC})

We constructed a custom WDDL_{SC} gadget, which encompasses the non-linear AND as well as the linear XOR operation to enhance the flexibility of the fabric. As discussed in Section 6.2.3, the desired gadget functionality is eventually programmed with the bitstream. Certain bits in the bitstream determine whether inputs or outputs are inverted by swapping the rails of their dual-rail representation (cf. Figure 6.3). In the case of our WDDL_{SC}, one additional configured multiplexer per output rail also selects the non-linear or linear function outputs. As the same configuration must apply for both rails, the same configuration bit is used inside the utilized CGB (cf. Figure 6.3).

Fault Propagation. The implementation consists of a WDDL_{SC}-AND that is inspired by the DPL-noEE [65] scheme, which was originally introduced to prevent the *early evaluation* effect. However, the original approach does not propagate all faulty inputs to the gadget outputs.

Lemma 3. *For any variable $x = (x_t, x_f)$ in a non-faulty DRP circuit, the gate $x_t \wedge x_f$ results in 0 and remains unchanged in both pre-charge and evaluation phases. If the function is combined with any other variable y using logical OR gate, the value of y is preserved at the*

output of the OR gate. In the presence of a fault, where $x = (1, 1)$*, the output of the OR gate is effectively tied to* 1*.*

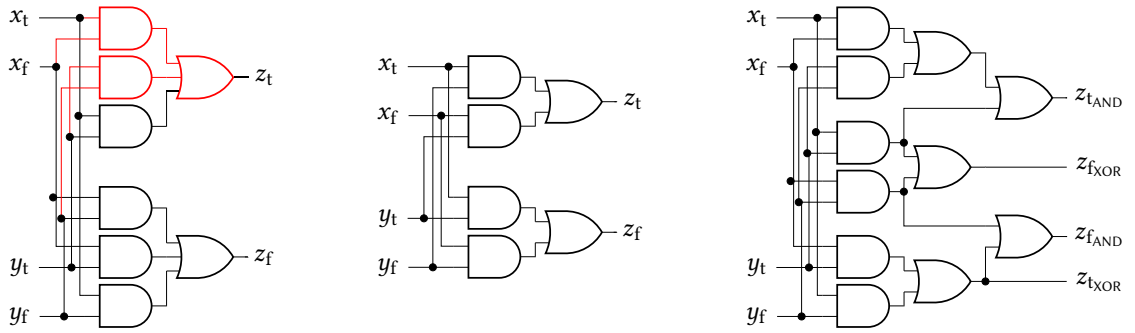
Following Lemma 3, we added two AND gates in the true rail computation of the WDDL AND gate to cover the cases where any input is faulted to $(1, 1)$ (cf. Figure 6.5(a)). Each of these gates has both rails of the same variable as their inputs, i.e., for the variable x the added AND gate computes $x_t \wedge x_f$ and for the input y the intermediate $y_t \wedge y_f$ is computed. Having $x = (1, 1)$ or $y = (1, 1)$ would then assure to have $z = (1, 1)$ at the output. Note that the case $x = y = (1, 1)$ is implicitly covered.

Lemma 4. *Considering any possible combination of a variable among* x_t, x_f *and another variable among* y_t, y_f *as two inputs of a logical AND gate, if at least one of the inputs* $x = (x_t, x_f)$ *or* $y = (y_t, y_f)$ *is* $(0, 0)$*, the output is assured to be* 0*.*

As the WDDL$_{SC}$ gadget implements an AND gate 'barrier' that considers every possible combination of input rails, a faulty dual-rail input $(0, 0)$ propagates, leading to the output $z = (0, 0)$. The case $x = y = (0, 0)$ is also implicitly covered. The architecture of WDDL$_{SC}$-XOR is shown in Figure 6.5(b) which – following the same concept – also assures fault propagation when any of the input is an invalid state $(0, 0)$ or $(1, 1)$. Due to the similar structure of WDDL$_{SC}$-AND and WDDL$_{SC}$-XOR, we combine them into the WDDL$_{SC}$-Combined gadget, illustrated in Figure 6.5(c), which provides both outputs and can naturally be given to a multiplexer configured to set the gadget's functionality. An analysis of every possible WDDL$_{SC}$-Combined input fault that is propagated to the outputs, reveals its functional correctness and proper fault propagation for every single faulty input and implicitly, the cases where $x = y$. The results are given in Table 6.2.

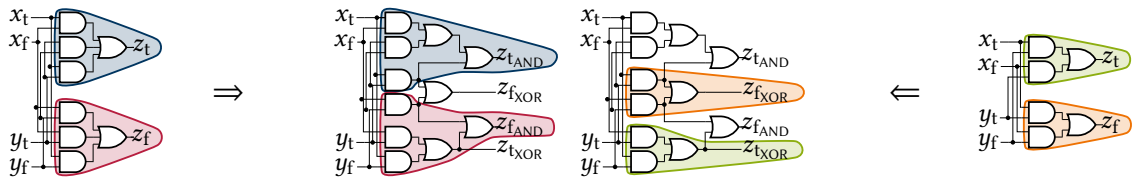
Lemma 5. *Since every WDDL*$_{SC}$ *gadget itself propagates an invalid input to its output, this fault propagation feature holds true for any arbitrary composition of WDDL*$_{SC}$ *gadgets.*

In a composition of WDDL$_{SC}$ gadgets, a faulty input encoding of one gadget is either directly faulted or originates from the output of a preceding gadget in the circuit. Consequently, the faulty dual-rail encoding propagates through such a composition. Since WDDL$_{SC}$ gadgets are only representing atomic functions and are not actual atomic gates, it must also be assured that faults at any gate within a gadget are either ineffective or lead to a faulty output. We refer to the general DRP circuit structure, which consists of two redundant parts – one computing the original value and the other its complement – and also applies to the internal circuit of the WDDL$_{SC}$ gadget. Since each gate within one of these parts contributes solely to the single-rail output computed by that part, a single fault can only affect that specific output rail, i.e., a fault within any DRP gadget cannot spread to multiple output rails (cf. Figure 6.6). As a result, the gadget always produces either a correct or an invalid dual-rail encoding, but never an incorrect yet valid dual-rail output. We verified this via exhaustive logic simulation, testing all possible inputs combined with injecting every possible single fault.



(a) WDDL$_{SC}$-AND. (b) WDDL$_{SC}$-XOR. (c) WDDL$_{SC}$-Combined.

Figure 6.5.: WDDL$_{SC}$ gadgets. The original DPL-noEE gates are highlighted in red.



(a) WDDL$_{SC}$-AND. (b) WDDL$_{SC}$-Combined. (c) WDDL$_{SC}$-XOR.

Figure 6.6.: Impact of internally injected faults on WDDL$_{SC}$ gadget output rails. Output rail contributing gates are shaded with color in both linear and non-linear gadgets and are transferred to the combined variant.

Lemma 6. *A single injected fault within a DRP gadget is either ineffective or leads to a faulty dual-rail encoding at the output.*

It is important to mention that the WDDL$_{SC}$ gadgets are explicitly not designed to be SCA secure. The WDDL$_{SC}$-Combined gadgets are only utilized in CGBs of the control-secure region of the fabric, i.e., their purpose is to compute values that are not masked. Each gadget must just properly propagate faulty inputs to its outputs and operate in the DRP protocol for compatibility with the rest of the fabric. Since the control-secure region and full-secure region are unidirectionally connected (discussed in detail later in Section 6.3.3), all modifications support the monotonic behavior in DRP circuits, i.e., we do not introduce any glitches that could propagate to the secure region and violate SCA security. For the same reason, we stick to the use of AND and OR gates inside the WDDL$_{SC}$ gadgets (instead of NAND and NOR gates) to ensure monotonic behavior for arbitrary gadget compositions.

6.3.2.2 Self-Checking LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL$_{SC}$)

The integration of LMDPL gadgets as a basic building block for the full-secure region requires proper fault propagation, while the formal security is preserved. We deploy a linear and a non-linear gadget, while the non-linear one is connected to the global PRNG that provides randomness in its dual-rail representation. An overview of the LMDPL$_{SC}$ which

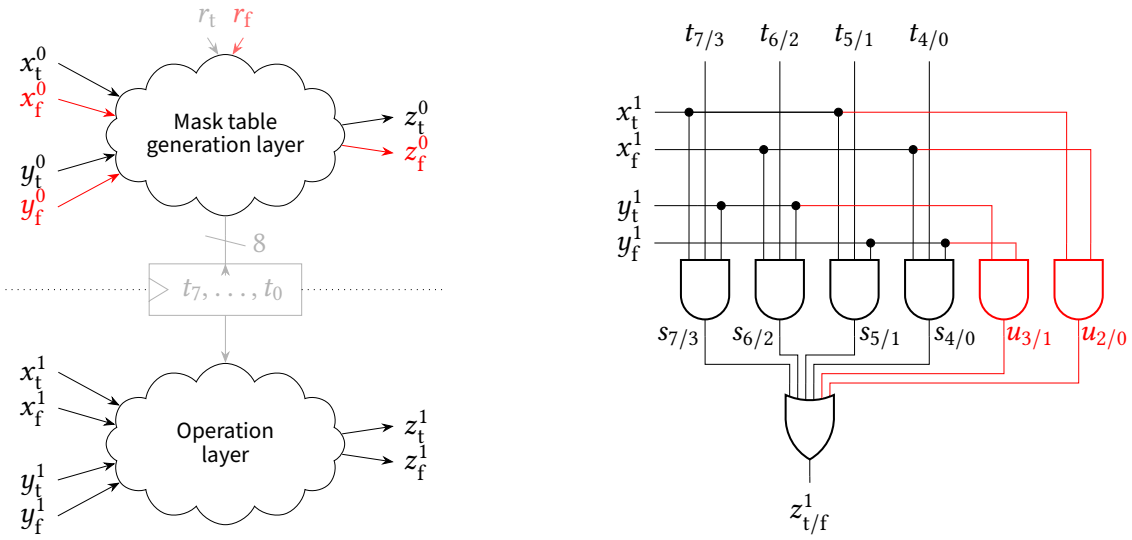
fault type	x_t	x_f	y_t	y_f	$z_{t_{AND}}$	$z_{f_{AND}}$		$z_{t_{XOR}}$	$z_{f_{XOR}}$	
none	0	1	0	1	0	1	✓	0	1	✓
	0	1	1	0	0	1	✓	1	0	✓
	1	0	0	1	0	1	✓	1	0	✓
	1	0	1	0	1	0	✓	0	1	✓
single input to $(0,0)$	0	0	0	1	0	0	✗	0	0	✗
	0	0	1	0	0	0	✗	0	0	✗
	0	1	0	0	0	0	✗	0	0	✗
	1	0	0	0	0	0	✗	0	0	✗
single input to $(1,1)$	0	1	1	1	1	1	✗	1	1	✗
	1	0	1	1	1	1	✗	1	1	✗
	1	1	0	1	1	1	✗	1	1	✗
	1	1	1	0	1	1	✗	1	1	✗
both inputs	0	0	0	0	0	0	✗	0	0	✗
	1	1	1	1	1	1	✗	1	1	✗

Table 6.2.: Covered invalid input propagation of WDDL_{SC}-Combined.

is a modified version of LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL) is shown in Figure 6.7(a).

Fault Propagation. Signals in the first share domain of the original LMDPL gadget (hence, inputs and output of the mask table generation layer) are not in dual-rail, and thus do not allow fault detection/propagation. Therefore, we replace such signals by their dual-rail counterparts to allow proper fault propagation based on the DRP encoding. This modification also goes hand in hand with our generic wiring approach, described in Section 6.2.4. Consequently, we apply alternating DRP phases in both layers. Because of the register placed between the layers of the non-linear gadget, the phases are carried out with one clock cycle offset. While intermediate values are evaluated based on x^0 and y^0 in the mask table generation layer, the operation layer is pre-charged. The intermediate result is stored in the register and further evaluated with shares x^1 and y^1 in the next clock cycle. During the evaluation phase in the operation layer, simultaneously, the mask table generation layer is pre-charged.

Proper fault propagation in the mask table generation layer of the non-linear variant is implemented by realizing every gate by its corresponding WDDL_{SC} gadget, previously explained in Section 6.3.2.1. Relying on Lemma 5, fault propagation of the mask table generation layer is assured. More precisely, a fault on x_t^0, x_f^0 or on y_t^0, y_f^0 always propagates to the intermediate values $t_7, \ldots, t_0$, which are also in DRP mode. A fault on the fresh random value (r_t, r_f) additionally propagates to the intermediate values $t_7, \ldots, t_0$ and to (z_t^0, z_f^0).



(a) LMDPL$_{SC}$ layer overview.

(b) Non-linear LMDPL$_{SC}$ op. layer for true/false rails.

Figure 6.7.: LMDPL$_{SC}$ with modifications compared to the original LMDPL in red.

The operation layer of the original LMDPL gadget propagates one or both invalid inputs $x^1 = (0,0)$ and $y^1 = (0,0)$ by default. Similar to WDDL$_{SC}$, Lemma 4 applies as no combination of input rails enables any AND gate to evaluate to logical '1' (cf. Figure 6.7(b)), resulting in the output $z^1 = (0,0)$. However, the invalid inputs $x^1 = (1,1)$ and $y^1 = (1,1)$ are not covered. Consider the following gadget inputs as an example, where x^1 is faulty.

$$x^0 = (0,1), \qquad y^0 = (0,1), \qquad x^1 = (1,1), \qquad y^1 = (0,1), \qquad r = (0,1)$$

Since the invalid input is located in the second share domain, the mask table generation layer pre-processes the intermediates based on the first share domain inputs as expected:

$$t_0 = t_1 = t_2 = t_7 = 0, \qquad t_3 = t_4 = t_5 = t_6 = 1.$$

In the mask table generation layer, the intermediate is 'selected' based on x^1 and y^1 values. More specifically, x_f^1 and y_f^1 enable the propagation of t_0 to s_0 and t_4 to s_4:

$$s_0 = s_1 = s_2 = s_3 = s_5 = s_6 = s_7 = 0, \qquad s_4 = 1.$$

Eventually, this leads to a valid dual-rail state at the gadget's output, despite the invalid input x^1:

$$z^1 = (0,1).$$

This example shows that an invalid input given to the LMDPL gadget may not be propagated to any of its outputs, and hence faulty but yet a valid dual-rail output is processed further. To alter this behavior, we extended the operation layer with two additional AND gates, following the same approach as for WDDL$_{SC}$ based on Lemma 3. Our modified

design is shown in Figure 6.7(b). Note that the signals are annotated for both rails allowing to show the circuit of both true and false rails. For example, $u_{3/1}$ denotes the true rail u_3 and the false rail u_1. With our modification, an invalid input $x^1 = (1, 1)$ leads to both $u_2 = 1$ and $u_0 = 1$ while $y^1 = (1, 1)$ leads to $u_3 = 1$ and $u_1 = 1$. Eventually, the occurrence of one or both of these invalid inputs always results in the invalid output $z^1 = (1, 1)$.

Besides the fault propagation of gadget inputs, injected faults inside the non-atomic gadget cause the internal circuit to behave similarly to that of WDDL_{SC} gadgets. In the event of a single injected fault within an LMDPL_{SC} gadget, the output either remains correct or leads to an invalid output encoding. Since the LMDPL_{SC} mask table generation layer is composed of WDDL_{SC} gadgets, Lemma 5 and Lemma 6 apply. As the operation layer is also a DRP circuit, Lemma 6 holds. We verified our claims through exhaustive logic simulation. Building on our fault propagation verification of single internally injected faults within a WDDL_{SC} gadget, we consider WDDL_{SC} gadgets as atomic gates. Our tests covered all possible inputs combined with every possible single fault injected.

Similarly, we follow the same procedure for the linear LMDPL_{SC}-XOR as with the non-linear mask table generation layer and replace every gate with its WDDL_{SC} counterpart. Due to the nature of the linear function, the layers operate independently. Hence, an invalid input in one layer results in an invalid output of that layer.

SCA Security. The authors of [193] showed first-order security of LMDPL non-linear gadgets under the SNI composability notion and glitch-extended probing model. In the original scheme, the mask table generation layer only operates on the single-rail representation of the first share of each variable, and the glitch propagation is stopped by the register in between the layers. In the argumentative approach, a contradiction between the glitch-extended probing model and the circuit behavior of the LMDPL operation layer is discussed. Essentially, glitch-extended probes assume leakage by the occurrence of glitches, which is never the case due to the proper execution of the DRP protocol that eliminates glitches entirely. The paper argues that even differences in timings at the second share domain do not reveal sensitive information due to the blinding of the generated intermediates. In the following, we apply and adapt the arguments for our modified variant.

Similar to the original scheme, it still holds that the mask table generation layer processes fresh randomness and the first share of each variable. Under the assumption that share domain separation is kept intact, generation of intermediates is independent of the second share domain and reveals no secret information. In our approach, we guarantee domain separation for arbitrary designs with connectivity constraints (cf. Section 6.2.4).

In the operation layer, we differentiate between intermediate and output probes to deduce glitch-extended first-order security under the SNI composability notion. Note that we do not cover combined attacks, i.e., SCA attacks on faulty circuits. Hence, considering a fault-free circuit, we focus on the gates/signals which we added to the original LMDPL gadget. If there is no invalid signal, $u_{2/0} = x^1_{\text{t}} \wedge x^1_{\text{f}}$ and $u_{3/1} = y^1_{\text{t}} \wedge y^1_{\text{f}}$ are tied to '0' in both pre-charge and evaluation phases with no toggles or glitches (cf. Figure 6.7(b)). Therefore,

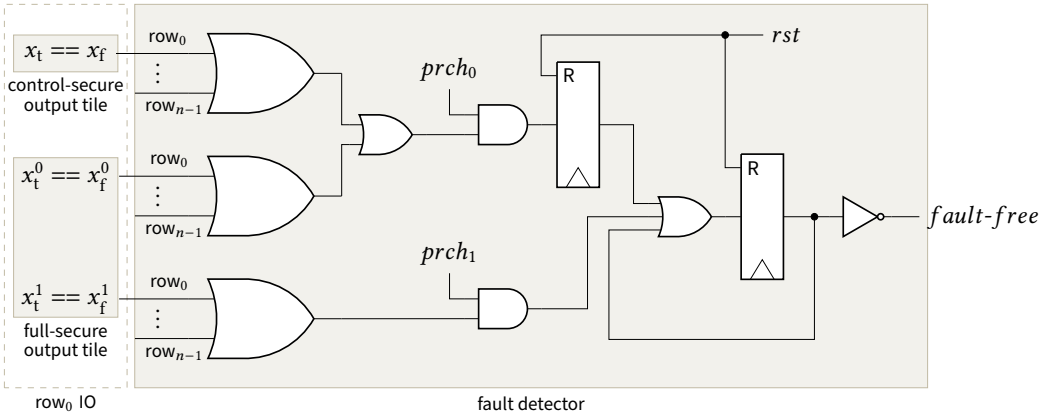


Figure 6.8.: The fault detector takes every XNOR of complementary rail outputs and computes a $fault\text{-}free$ signal. Upon fault detection, the signal toggles to 0 and is maintained in a register until the device is reset. Inside each output tile, the boolean AND function is applied to the output and the $fault\text{-}free$ signal, effectively setting every primary output to logical zero if a fault is detected.

a probe placed on any of these wires does not reveal any information. Moreover, the original security analysis of LMDPL argues that a probe placed on the output z_t^1 (or z_f^1), which observes '1', may reveal information about timing differences. As $u_{2/0}$ and $u_{3/1}$ never toggle in a fault-free circuit, no additional timing information is issued at the output. Thus, the original security argument by the original authors stays valid.

6.3.2.3 Fault Detector

Building on the fault propagating WDDL$_{SC}$ and LMDPL$_{SC}$ gadgets, it is sufficient to connect a fault detector module to all primary outputs of the secure fabric, including both types of regions: full-secure and control-secure. Since all gadgets integrate self-checking properties for fault propagation, a fault at any gadget input will propagate through the gadget itself and through the consecutive gadgets to at least one primary output, where it can be detected. As shown in Figure 6.8, the detection circuitry employs an XNOR gate for every dual rail of primary outputs, which identifies an invalid dual-rail state in the evaluation phase. In case one rail is faulty, the XNOR results in logical '1' and is forwarded through an OR-tree that covers the XNOR result of the dual rails of the same share domain. The first share domain also covers the XOR of the control-secure region. Since the share domains are evaluated with one clock cycle offset (mask table generation layer versus operation layer, see Figure 6.7(a)), the corresponding individual pre-charge signals are utilized to activate the detection only in the respective evaluation phase of the share domain, i.e., $prech_0$ and $prech_1$ signals in Figure 6.8. By this structure, domain separation is kept intact in the fault detector module to not violate the SCA requirements. As long as no faults are detected, the $fault\text{-}free$ signal is set to 1 and remains stable. If any of the dual-rail primary outputs is in an invalid state, the $fault\text{-}free$ signal toggles to 0 and is maintained in a register until the fabric is reset. Inside each IO tile, including both full-secure and control-secure regions, the $fault\text{-}free$ signal is ANDed with the output signal to effectively avoid issuing any output if a fault is detected until the device is reset.

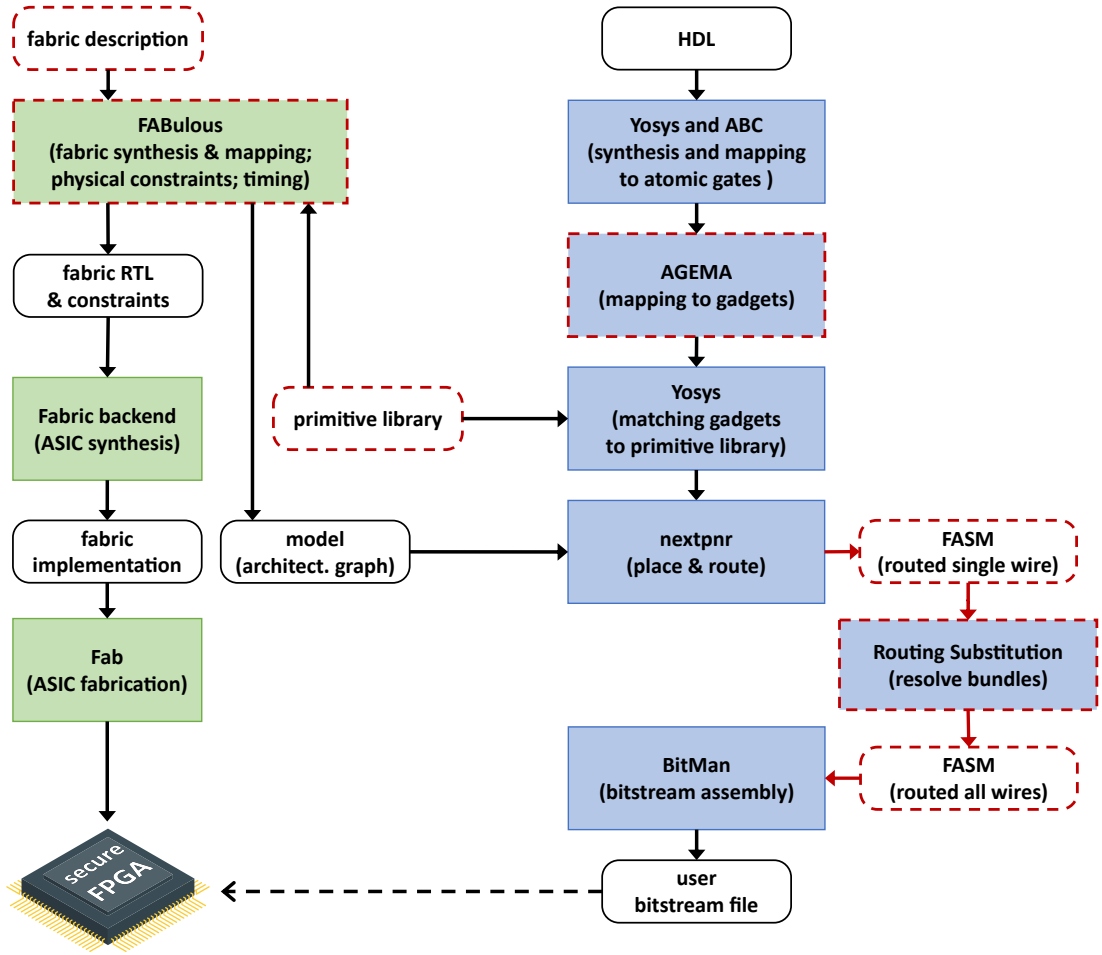
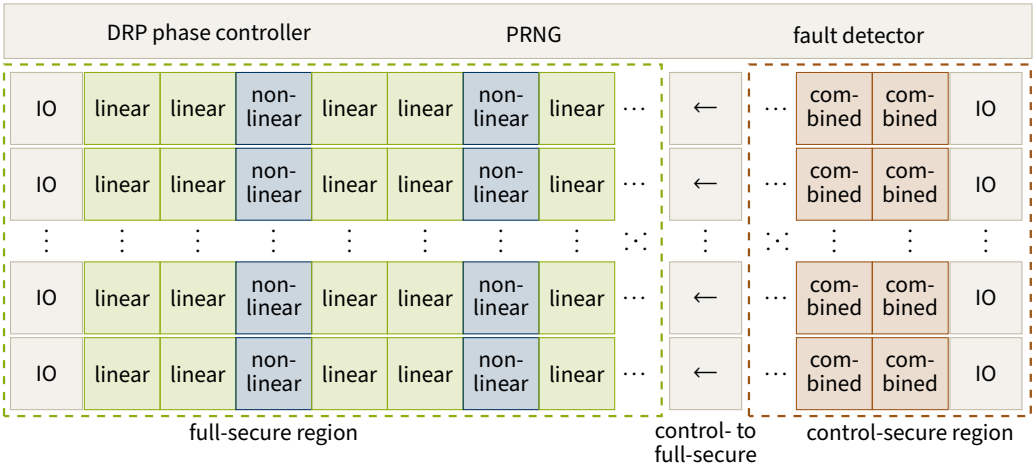


Figure 6.9.: Design flow of a reconfigurable fabric using the FABulous framework. Red highlighted steps show extensions and modified tools.

6.3.3 Secure Fabric Implementation using FABulous

Since FABulous was not designed for our particular purpose, modifications and extensions to its framework were necessary. The main modifications are the change of basic primitives from LUTs to secure gadgets and configurable wiring adjustments to use only the aforementioned parallel switch matrices. In addition, modules for fault detection, fresh randomness supply, and control signal generation to control DRP phases are integrated. Those modules are automatically scaled to fit the fabric size, including their fixed wiring to the corresponding secure fabric CGBs. After our adjustments, the fabric is easily scalable as in the original FABulous framework. A table specifies the number of rows and columns in the fabric, along with the CGBs types assigned to each cell. In this way, the size of the fabric, the full-secure and control-secure regions as well as the ratio of linear and non-linear CGBs are specified. The general flow for the fabric implementation is illustrated at the left side of Figure 6.9.

Initially, a description of the fabric is required, which consists of two main parts. The first is a table that defines fabric organization tile by tile, which also contains a detailed



(a) Secure fabric layout.

Region	Type	Gadget	Data	#Wires	#Shares	DRP	Fault Propagation	Provable Side-channel Secure
full-secure	linear	LMDPL_{SC}-XOR	secret	4	2	✓	✓	✓
	non-linear	LMDPL_{SC}-AND						
control-secure	combined	WDDL_{SC}-Combined	public	2	1	✓	✓	✗

(b) CGB type classification overview.

Figure 6.10.: Tile layout in our secure reconfigurable fabric case study. The fabric is partitioned in two security regions with accordingly equipped CGBs, indicating their functionality. Global module extensions are listed at the top.

description for each tile. This covers utilized primitives, available local wires inside the tile, wires for interconnections to neighbored tiles, and a reference to the switch matrix used. Second, each tile is additionally represented by a set of files. HDL files define the primitives, such as reconfigurable secure gadgets or IO, a switch matrix description, which corresponds to a PSM in our case, and a description of local wiring inside the tile. Therefore, not only primitives may differ from tile to tile, but also switch matrices, allowing for different kinds of routing, such as for dual-rail only (control-secure region) or for masked dual-rail signals (full-secure region).

Since we implement custom primitives and do not implement the basic primitives available in FABulous, the primitive library is also adjusted. All unused primitives are removed and the design abstractions for every deployed gadget type in our CGBs are added. This step is needed to enable the automated flow of mapping HDL designs to the fabric. Using the fabric description and the primitive library, FABulous generates RTL files for the secure fabric, which are then used for synthesis and fabrication. In addition, FABulous generates an architectural graph model, which is needed for place-and-route of the designs mapped to the fabric by nextpnr.

LMDPL$_{SC}$- and WDDL$_{SC}$-based Secure Fabric. Recalling our defined full-secure and control-secure regions, the non-secure parts of the system are out of the scope, since we assume a secure fabric to be embedded into an SoC. Figure 6.10 abstracts the structure of our case study implementation. On the left side of the fabric, we placed the full-secure region, which includes full-secure IO ports, full-secure non-linear logic CGBs, and full-secure linear logic CGBs based on LMDPL$_{SC}$ gadgets.

The fresh randomness for non-linear LMDPL$_{SC}$ gadgets is provided by a PRNG every other clock cycle during the evaluation phase of the mask table generation layer. We integrate an unrolled design of the Trivium stream cipher, which is scalable with the fabric size [194]. Every gate in our Trivium implementation is substituted with its WDDL$_{SC}$ counterpart to also ensure fault propagation during randomness generation. In the scope of this implementation, we expect a seed generated by a TRNG to be loaded into the PRNG only at power-up cycle. The PRNG outputs have a fixed wiring, such that each random bit is uniquely provided to a single non-linear gadget.

On the right side, the control-secure region is placed, which includes control-secure IO ports and WDDL$_{SC}$-Combined gadget-based CGBs to realize linear and non-linear dual-rail but not masked logic. Between these regions, we place a tile column to interconnect them. The connection is implemented unidirectional, namely from control- to full-secure only. This is done by concatenation of an artificial share tied to '0', i.e., showing x by $(x^0 = x, x^1 = 0)$, more precisely by $(x_t^0 = x_t, x_f^0 = x_f)$ and $(x_t^1 = 0, x_f^1 = 1)$. Connection in the other direction is not allowed, as it would pose a security breach (see Section 6.2.4).

To enable fault detection, all IOs perform a check of the valid dual-rail state if configured as output. The resulting signals are hardwired to the fault detector module, which stores a fault detection indicator in a register. A detected fault in the DRP evaluation phase sets a dedicated alarm output signal, and the output ports are simultaneously disabled. The PRNG, the controller of the DRP phases, and the fault detector are adjustable during fabric design and are fixed in the fabricated device, i.e, they are implemented similarly to an ASIC without being reprogrammed after fabrication.

6.3.4 Open-Source Toolchain for Bitstream Generation

The complete flow to generate the configuration bitstream file for a given HDL design is shown in the right side of Figure 6.9. The process starts with an unprotected annotated HDL design provided by the user that should be automatically secured with respect to SCA and FI attacks. This design can be any algorithm or function and is mapped to the atomic logic gates by Yosys in the first step.

AGEMA determines the allocation of wires, registers, and gates to their corresponding security region based on primary input annotations by the end-user in the unprotected HDL design, classified as either secret or public. The framework sees the given circuit as a Mealy machine and handles the primary input annotations for secret data by propagating them through the circuit. The remaining parts belong to the control logic that does not

depend on secrets and is not masked. In case the control logic operates on secret data, the masking is also applied in that part of the circuit.

Usually, cascaded gadgets of a gadget-based masking scheme contain register stages and lead to extra clock cycles compared to the given original design. LMDPL / LMDPL_{SC} maintains its constant cycle count also in sequential logic, e.g., the state computation of an FSM. This eliminates the need for additional handling or balancing of control signal delays with the rest of the circuit.

Afterward, AGEMA is used to map the atomic logic gates to the deployed fabric primitives, i.e., secure reconfigurable gadgets. To be able to translate the atomic logic gates to the custom gadgets in the proposed fabric, we extended the tool AGEMA to support LMDPL_{SC} and WDDL_{SC} gadgets and generate protected netlists using our primitives only. In the translation process, full-secure non-linear and linear gadgets as well as control-secure gadgets are instantiated. The parameters of the instantiated Verilog modules are set to achieve the desired functionality inherent to the atomic gates they represent. These parameters equal the configuration bits determined by the bitstream that is later generated and programmed onto the device.

The result of AGEMA is a Verilog netlist, which uses parameterized Verilog modules for secure gadgets. At this state, the dual-rail signals as well as masking shares are bundled and represented by a single wire. Any wiring to distribute fresh random bits is also neglected, as randomness is already available to each non-linear masked CGB and does not need to be included in the reconfigurable routing.

In the next steps, we follow the adjusted F4PGA flow with *bundle-routing* without consideration of randomness generation or distribution. Yosys is used again to generate the JavaScript Object Notation (JSON) formatted mapped netlist needed for the nextpnr tool. This step is trivially achieved, since the netlist from the previous synthesis already uses target primitives, and no optimizations are done in this step. Then, the architectural graph model generated by the FABulous framework and the JSON formatted mapped netlist are used for place-and-route with nextpnr. The intermediate result is an FPGA Assembly (FASM) file that describes the placed and routed netlist.

Until this point, the complete workflow treated the dual-rail and masked dual-rail signals as single wires, i.e., bundle-routing. A custom script converts the intermediate netlist to "resolve" the wire bundles by inserting all missing wires directly into the FASM file. In this way, the parallel routing is achieved with minimal overhead based on share domains and rail indicators.

Eventually, the routed netlist with all wires is used by BitMan to create a user bitstream file in the specified format to program the customized reconfigurable secure fabric.

6.3.5 Overhead Comparison

Before diving into the conducted overhead comparison, it should be noted that any protection mechanism comes with its own overheads, such as area, energy, and delay. Therefore,

Table 6.3.: Comparison of area required to map different cryptographic implementations to a secure or conventional fabric with our proposed scheme. The comparison is based on the estimated Gate Equivalent (GE) required to implement the tiles including primitives, switch matrices and configuration memory.

	AES-128 (byte-serial) [196]	**AES-128** (round-based) [196]	**Midori-64** [197]	**CRAFT** [198]	**Keccak** -f[200] [199]
Latency [cycles]	454	22	34	64	1800
Random [bits/cycle]*	389	660	256	192	200

*Numbers stated on average. Double the amount is required during the evaluation phases (every other cycle).

Default FABulous fabric

Cipher [GE]	15453 k	42271 k	13470 k	9364 k	12722 k
#CLBs	2416	6608	2105	1463	1988
#LUT4	19328	52867	16841	11707	15905
#FF	5876	10576	3600	3108	4036
PRNG [GE]	19585 k	32581 k	13201 k	10131 k	10515 k
#CLBs	3062	5094	2064	1584	1644
#LUT4	24492	40752	16512	12672	13152
#FF	1152	1152	1152	1152	1152
Total [GE]	35038 k	74852 k	26671 k	19495 k	23237 k

Proposed secure fabric

Cipher [GE]	4648 k	11124 k	3851 k	1963 k	3617 k
difference	*- 70%*	*- 74%*	*- 71%*	*- 79%*	*- 71%*
#linear	998	3088	944	375	1004
#non-linear	778	1320	512	384	400
#control	116	36	143	24	73
PRNG [GE]	158 k	266 k	105 k	79 k	82 k
difference	*- 99%*	*- 99%*	*- 99%*	*- 99%*	*- 99%*
Total [GE]	4806 k	11390 k	3956 k	2042 k	3699 k
difference	*- 86%*	*- 85%*	*- 85%*	*- 89%*	*- 84%*

a secure reconfigurable fabric is certainly larger and consumes more energy compared to an unprotected reconfigurable platform. However, a fair comparison would be to compare with the same secure design implemented on a conventional fabric.

To demonstrate the efficiency of our proposed method and the gadget-based fabric, we compare it with the LUT-based FABulous reference implementation, synthesized with the same toolchain. This implementation is similar to many commercial FPGA architectures, since FABulous provides a default template with LUT4 based Configurable Logic Blocks (CLBs), which we use without any modifications. Each tile with such a CLB consists of 8

Table 6.4.: Comparison of ISCAS89 benchmark synthesis results: overhead as Gate Equivalent (GE) between secure and conventional reconfigurable fabrics. The unprotected designs serve as baseline.

	original	**FABulous fabric**			**Our proposed fabric**	
	ASIC	**unprot.**	**WDDL$_{SC}$**	**LMDPL$_{SC}$**	**WDDL$_{SC}$**	**LMDPL$_{SC}$**
s27	19	6396	19188	191880	10048	20834
	-99%	-	*+200%*	*+2900%*	*+57%*	*+226%*
s382	195	38376	198276	2078700	118064	244286
	-99%	-	*+416%*	*+5316%*	*+207%*	*+536%*
s420	209	31980	185484	1477476	108016	219660
	-99%	-	*+480%*	*+4520%*	*+238%*	*+587%*
s641	216	63960	236652	2890992	163280	338908
	-99%	-	*+270%*	*+4420%*	*+155%*	*+430%*
s713	216	63960	243048	2890992	162024	336442
	-99%	-	*+280%*	*+4420%*	*+153%*	*+426%*
s1238	722	166296	633204	9721920	536312	1115392
	-99%	-	*+281%*	*+5746%*	*+222%*	*+571%*
s1423	802	140712	761124	7483320	473512	972658
	-99%	-	*+441%*	*+5218%*	*+236%*	*+591%*
s1488	776	204672	709956	11647116	630512	1314088
	-99%	-	*+247%*	*+5590%*	*+208%*	*+542%*
s5378	1847	332592	1650168	17736108	1170592	2406700
	-99%	-	*+396%*	*+5233%*	*+252%*	*+623%*
s9234	1375	243048	1387932	13585104	1562464	3207376
	-99%	-	*+471%*	*+5489%*	*+543%*	*+1219%*
s13207	3888	543660	3479424	32945796	2505720	5119856
	-99%	-	*+540%*	*+5960%*	*+361%*	*+842%*
s15850	4682	812292	4816188	45565104	3158840	6481650
	-99%	-	*+493%*	*+5509%*	*+289%*	*+698%*
s35932	15936	2315352	11256960	98562360	5935856	12206842
	-99%	-	*+386%*	*+4157%*	*+156%*	*+427%*
s38417	12888	2296164	14512524	135313776	9771680	19967896
	-99%	-	*+532%*	*+5793%*	*+325%*	*+769%*
s38584	15230	2417688	15369588	163161960	11677032	24035562
	-99%	-	*+536%*	*+6649%*	*+383%*	*+894%*
avg.	*-99%*	-	*+398%*	*+5128%*	*+252%*	*+625%*

Table 6.5.: Area estimate reference for handcrafted first-order secure cryptographic designs synthesized as ASICs and mapped to the FABulous reference fabric.

Implementation	**ASIC** [GE]	**FABulous** [GE]	**RNG** [bit]	**Latency** [cycle]
AES [200]	9513	1375140	54	276
AES [201]	9141	1349556	18	246
AES [202]	15766	2347332	48	266
AES [203] v1	7437	991380	4	236
AES [203] v2	12695	1874028	8	148
CRAFT [204]	9265	1042548	0	64
Midori-64 [205]	10015	1029756	0	32
Midori-64 [206]	9881	1432704	0	32

instances of LUT4, registers, a switch matrix, and configuration memory. Compared to that, in our fabric design one tile consists of a CGB with a configurable gadget, registers, a PSM and configuration memory. For comparison, the same HDL designs are used on both platforms, as follows. In case of the conventional fabric, the HDL is first translated into a protected netlist based on the same security scheme using AGEMA. We labeled the primary key and plaintext inputs as secret data and primary control inputs as public, which leads to a mapping of the control logic to the control-secure region, while the logic processing secret data is allocated to the full-secure region. The resulting secure designs are mapped to fabric based on (1) LUT4 or (2) reconfigurable secure gadgets. Estimation of area differences is conducted by the GE required to implement tiles with CLBs or CGBs, respectively, including switch matrices and configuration memory. The results are shown in Table 6.3.

It should be noted that remaining free resources of a secure fabric can be utilized for some non-secure logic, if necessary. However, due to the overhead of the security measures in the secure fabric, a conventional FPGA would utilize free resources much more efficiently by not implementing any security measures in such a case.

In Table 6.4 we extend our overhead comparison with non-cryptographic ISCAS89 benchmark circuits. The baseline for each comparison is the unprotected benchmark circuit mapped to the LUT4 CLBs of FABulous fabric. These results are then compared to the original ASIC circuit, and protected versions mapped to the original FABulous fabric and our proposed secure fabric. Since this table is created to show the overheads, we distinguish between two options, where the whole circuit is mapped either to the control-secure gadgets ($WDDL_{SC}$) or the full-secure gadgets ($LMDPL_{SC}$). For the $LMDPL_{SC}$ gadgets, we do not consider the overhead required for the PRNG. In the case of the unrolled DRP version of the Trivium cipher (for the PRNG), our proposed secure fabric requires less than 1% of the area needed by the conventional fabric.

To provide a comprehensive evaluation, we include area estimates for handcrafted masked first-order secure implementations of AES, CRAFT and Midori-64 ciphers in Table 6.5. For fair comparison, we synthesized these designs as ASICs, as well as mapped them to the FABulous reference fabric implementation. For both procedures, we used the same

open-source tools, namely Yosys and ABC. Notably, not all handcrafted designs can be directly mapped to an FPGA, such as the FABulous reference fabric and may need some adjustments, e.g., due to the lack of specific registers with asynchronous reset. Due to the nature of reconfigurable fabrics, the overhead compared to ASIC implementations is significantly higher for such mapped designs. Even mapped to a reconfigurable fabric, those handcrafted designs introduce significantly less overhead compared to the proposed gadget-based secure fabric, which can be followed based on the combination of results in Table 6.3 and Table 6.5. However, handcrafted implementations are individually designed, while the proposed fabric is capable of protecting arbitrary designs.

6.3.6 Experimental Evaluation

We verified the correctness of the proposed primitives and architecture by logic simulation with and without fault injection. To illustrate the practical relevance of our approach, we conducted SCA experiments with a real FPGA device. We emulated a small secure fabric on a real platform and programmed it with a generated bitstream.

6.3.6.1 Logic Simulation

To evaluate the proposed secure fabric, a series of logic simulations using Icarus Verilog was conducted. Initially, exhaustive simulations on individual gadgets of every type were performed to verify correct functionality and assess the fault propagation property. In this scope, we tested the propagation of faulty inputs and single faults within each gadget for every possible input combination. Subsequently, we simulated a small fabric configured with simple circuits, such as a small MUX tree, which was exhaustively tested for functional correctness. Special emphasis was placed on single-bit fault injection at every possible gadget input. Notably, our results demonstrate that even if a faulty dual-rail state occurs in an inactive circuit path, e.g., at an unselected MUX input, the outputs consistently enter an invalid dual-rail state. Finally, we simulated a fabric of sufficient scale to implement a round-based AES, including the complete fabric configuration process with a given bitstream. This fabric's correct functionality was validated for a variety of ciphers with according bitstream configurations, including LED, Skinny, Midori-64, CRAFT, round-based and serial AES implementations.

6.3.6.2 Setup

Our platform of choice is a Xilinx Kintex-7 FPGA placed on a SAKURA-X board [189]. We collected power traces by monitoring the voltage drop over a 1 Ω shunt resistor on the Vdd path of the operating FPGA. The platform is clocked by a stable 6 MHz oscillator, while a digital oscilloscope samples the power consumption at a frequency of 500 MS/s. The security analysis relies on the known and common fixed versus random t-tests [59].

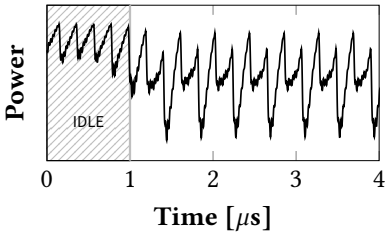
Due to the high emulation overhead and resulting resource constraints of the physical platform, we deploy a target design that contains a secure fabric with 23×7 full-secure and 23×1 control-secure CGB tiles. This configuration provides a total of 161 full-secure tiles with a 2-to-1 ratio of non-linear and linear gadgets, 23 control-secure tiles, and 23 IO tiles per region. The selected size of the emulated FPGA is sufficient to run an AES S-Box with minimal control logic to execute it in a loop. More precisely, the given input is fed into the AES S-Box, and then its output supplies its input in the further clock cycles.

After the physical device is programmed with the target bitstream, the measurement process is started. In the beginning of that process, the bitstream for the emulated design is serially sent to the target device and serially programmed onto the emulated secure fabric once. In addition, a random seed for the integrated PRNG is provided. For each power trace measurement, the emulated device excluding the PRNG is reset to the default state and restarted. It means that the random seed is only provided and loaded only once right after programming the bitstream. Further, we made sure that the DRP phases of the PRNG are synchronized with the full-secure region.

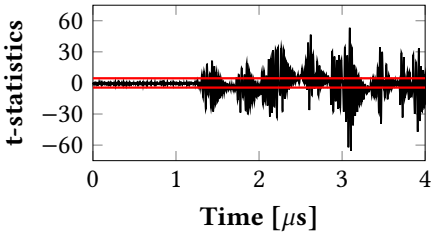
Utilization. Despite the small emulated fabric size, a 52% LUT utilization of the Kintex-7 is reached by our target design. 99% of which belong to the emulated FPGA. The PRNG takes up a share of about 5.29%, while this ratio should be smaller for larger fabric sizes as the PRNG is also emulated by DRP logic on this commercial FPGA. The resources used by the other global modules, e.g., the fault detector, are comparably negligible. Upon closer inspection of the utilization, the LUT requirements for the emulated switch matrices stand out. As the reconfigurable wiring is realized by multiplexers (FABulous default), the realization with LUTs takes about 85% of the emulated device. For experimental SCA evaluations, we take into account the high limitations that follow from this and program a bitstream that implements a single looped AES S-Box onto the emulated device. In a manufactured secure FPGA, PSMs would be implemented much more efficiently by logic gates instead of LUTs. However, in our emulation setup, 94% of the full-secure region and 82% of the control-secure region are occupied.

6.3.6.3 Results

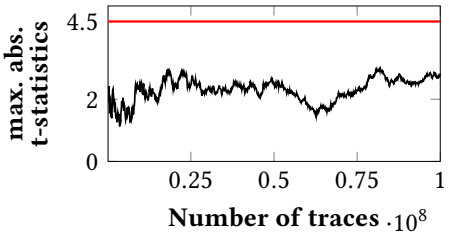
For the security analysis, we supplied the design with either a fixed or random input in a random sequence. In both cases, the input is masked using independent and fresh randomness. After a few idle cycles, as visible in Figure 6.11(a), the emulated device starts, infinitely the execution of the looped AES S-Box. A first-order fixed vs. random t-test over 100 million traces does not show signs of first-order leakage. The change in the maximum absolute t-value over the number of traces in Figure 6.11(c) remains clearly below the threshold of 4.5. Figure 6.11(d) and confirms this for the t-value of every sample point in the power trace after 100 million traces. As a countercheck, we repeated the same experiment without fresh randomness, i.e., we turned the integrated PRNG off. Already after 1 million traces, the t-test shows a significant leakage, plotted in Figure 6.11(b). Furthermore, we conducted a second-order fixed vs. random t-test as the sanity check to examine the



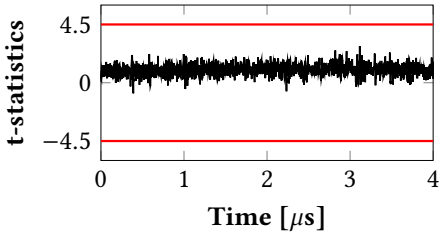
(a) A sample trace.



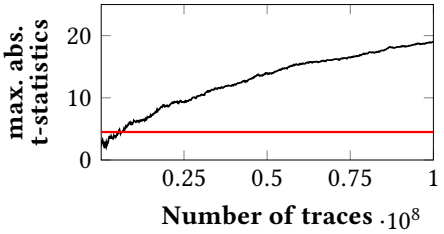
(b) First-order t-test, without fresh randomness, 1 million traces.



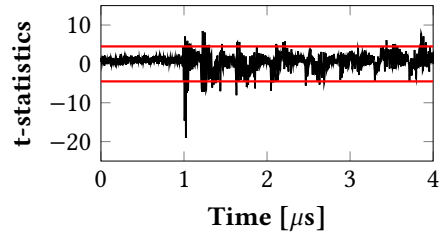
(c) First-order maximum absolute t-statistic, with fresh randomness.



(d) First-order t-test, with fresh randomness, 100 million traces.



(e) Second-order maximum absolute t-statistic, with fresh randomness.



(f) Second-order t-test, with fresh randomness, 100 million traces.

Figure 6.11.: Experimental analysis results, including a sample trace and univariate fixed vs. random t-test results.

detectability of higher-order leakages. According to our expectations, the experiment shows second-order leakage (cf. Figure 6.11(e) and Figure 6.11(f)). The conducted practical experiments reinforce the general approach, theoretical assumptions, and our concrete implementation.

6.4 Discussion

In this section, we critically discuss our methodology and results. We compare the conventional general-purpose approach with the proposed approach tailored to integrated physical security.

6.4.1 Secure Fabric Compared to Conventional FPGAs

Our fabric as well as conventional FPGAs both provide the ability to reconfigure the hardware to meet specific requirements of the use case after the manufacturing process. Both of these devices utilize some fixed primitives such as LUTs or secure gadgets, and switch matrices. However, our fabric implements its primitives securely with respect to the selected scheme and the corresponding overhead. Additionally, it implements several submodules such as RNG and fault detector, which are hard-wired. Therefore, it is quite effective in realizing the selected security scheme. However, our approach solely relies on the security assumption of the underlying scheme, which itself remains unchanged after fabrication. A conventional FPGA is more flexible in these terms and is better suited to implement different schemes, but compromises security and efficiency, on the other hand. For example, it would usually require implementing everything with the same reconfigurable primitives, including modules such as an RNG or fault detector. This is not only less efficient compared to ASIC modules but also utilizes non-optimized switch matrices for wiring, which introduces additional overheads.

In our case study, the selected gadgets provide protection against first-order SCA attacks and enable fault detection as a countermeasure against some FI attacks. However, it should be highlighted once more that our proposed methodology can be applied to any kind of secure gadgets. For example, higher-order SCA-secure gadgets can be selected to construct CGBs of the secure region. This would naturally lead to higher number of share domains, etc. In the same way, gadgets may implement different approaches for fault detection or other measures against fault attacks like fault correction. It is also possible to implement gadgets with SCA resistance or for fault detection only.

Additionally, it should be noted that optimizations can be done to both LUT-based and gadget-based fabric tiles with respect to the target requirements, such as the security schemes employed. However, since our main goal is to show an approach for secure reconfigurable fabric design and to evaluate them from a security perspective, optimizations are considered as lower priority. For example, we use only one secure gadget per CGB, which is sufficient for the evaluation, and makes the analysis process less complex and easier to understand at the same time.

The primary constraint for the proposed approach remains the fabric size, which directly limits the complexity of designs that can be accommodated. According to our synthesis results, we estimate that a round-based AES cipher implementation requires a proposed secure fabric with circuit complexity comparable to a conventional FPGA fabric, such as FABulous, with 15k LUT4. Typically, symmetric crypto implementations (e.g., AES) can be converted to a Boolean masked circuit efficiently and benefit from this approach. Nevertheless, not all algorithms are well-suited for the application of Boolean Masking, e.g., many add–rotate–XOR (ARX) and Post-quantum cryptography (PQC) algorithms, and are at least highly inefficient or even practically unreasonable due to the required fabric sizes.

6.4.2 Use Cases

In this section, we would like to highlight use case scenarios of our proposed physically secure reconfigurable fabric. In general, it can be used whenever SCA and/or FI attacks pose a risk to the system. Changes and/or security patches can be easily applied via the reconfigurability nature of the fabric. This makes it beneficial for use in new or fast-evolving products, as well as very flexible for integration into the existing systems.

In order to map an HDL design to the secure fabric, no specific knowledge about the details of the secure fabric or the security schemes is required. The tool-chain fully automates this process including mapping, placement, and routing of the target design, according to the implemented security measures. This can greatly reduce the development efforts as well as the risks of human-induced errors when applying security measures.

Our proposed hardware is ideal for systems requiring enhanced security due to the processing of very sensitive information, such as in financial or medical data. It is also crucial for critical infrastructure or safety-critical systems, where any malfunction or interruption could result in significant damage and/or pose substantial risks to human life and property.

Several more detailed examples of applicable use cases can be found below:

- Cryptographic Devices

  Hardware Security Modules (HSMs) and secure elements used for cryptographic operations (e.g., key generation, encryption, decryption, and digital signatures).

- Automotive

  Electronic Control Units (ECUs), CAN bus and protocols, keyless entry systems, and in-vehicle infotainment systems which require robust security to protect against unauthorized access and ensure the safety of passengers.

- Industrial Control Systems

  Systems used in critical infrastructure sectors such as energy, water, manufacturing, and transportation which must be protected against cyberattacks that could disrupt operations or compromise safety.

- Payment Systems

  Secure payment systems such as Point-of-Sale (POS) terminals and contactless payment cards which handle sensitive financial data, such as credit card information and Personal Identification Numbers (PINs).

- Trusted Platform Modules (TPMs)

  TPMs used in personal computers, servers, and mobile devices to provide hardware-based security functions such as key storage, digital signatures, and secure measurements.

- Boot and Firmware Integrity

  Devices that use secure boot mechanisms to verify firmware integrity, such as IoT devices, smartphones, laptops, and automotive control units.

- Medical Devices

  Connected medical devices used in hospitals and clinics which must protect sensitive health data and ensure patient safety.

- Military and Aerospace Systems

  Embedded systems in military hardware, satellites, drones, and secure communications equipment.

6.4.3 SCA Resistance

LMDPL has already been evaluated multiple times in the literature and its first-order security has been proved both in theory and in practice [45], [76], [77], [207]. Estimating higher-order statistics (necessary to conduct higher-order attacks on masked implementations) is susceptible to low SNR, since the required number of SCA traces increases exponentially with the noise level [208]. As a reference, the authors of [45] have shown practical (not theoretical) robustness of an LMDPL AES round-based design against second-order attacks. Therefore, we expect a high level of difficulty to successfully conduct higher-order attacks on our fabric due to the realization of every signal of LMDPL_{SC} with a DRP scheme, hence a lower SNR (see the construction of LMDPL_{SC} given in Section 2.4.4). Note that the first share domain of LMDPL gadgets are implemented by single rail logic while this is not the case in LMDPL_{SC} gadgets.

Nevertheless, we should comment on a low number of traces required to detect a second-order leakage in our experiment, i.e., Figure 6.11(e). This is due to the fact that we used a commercial FPGA to emulate our fabric. It means that several LUTs and switch boxes of the employed Xilinx FPGA are involved in every DRP circuitry which makes them away from being able to reduce the SNR. More precisely, the number of logic elements involved in dual rails are not balanced, resulting in not only imbalanced routing but also potentially imbalanced number of toggles. This is not the case when the fabric is implemented as an ASIC chip, and we expect to see considerably lower amount of second- and higher-order leakages.

6.4.4 FI Resistance

Preventing fault attacks is very costly, if even possible, and in combination with proper SCA resistance, such as masking, it becomes even more challenging. In our approach, we focus on the self-checking principle and provide a sound detection mechanism for faults. As explained in Section 6.2.6, fault injection attempts additionally affect some untargeted signals/gates and cause some invalid dual-rail states, which would be sufficient to detect

fault injection in our approach. Therefore, our approach can detect faults which may stay undetected by schemes comparing only redundant primary outputs, e.g., majority voting. It should be noted here that faults can be detected in all paths of the entire circuit simultaneously, including those which do not directly affect the primary outputs. This implicitly covers setup time violation attacks such as FSA [21] and does not require additional countermeasures as in [209] due to the propagation of faults to the primary output even if it is occurring in a not activated path. To elaborate on this, it is true that we examine the validity of dual-rail outputs only, but as explained in Section 6.2.6, any invalid state in the entire circuit would stop the proceeding circuit to generate a valid dual-rail state. Therefore, if a fault is injected and all dual-rail outputs are in a valid state, the fault was not effective or it has been injected after the primary outputs have delivered the circuit output. Note that even in such a case, invalid dual-rail states will be stored in the register, and they reach the primary output in the next clock cycle.

Theoretically, it is possible to inject multiple faults in such a way that the resulting signals are in a valid dual-rail state and bypass our detection facility. However, the rails associated to a logical signal are always placed closely to each other due to our routing constraints, which guarantees that dual-rail pairs are routed together. In addition, the challenge is to cause different faults, one rail from 0 to 1 and the complementary rail from 1 to 0, at the same time, while the targeted wires are placed close to each other. Therefore, we assume that even highly localized fault injections equipments have difficulties to inject undetectable faults in our fabric.

6.4.5 Other Attack Vectors

In this work, we focused on the security of the designs mapped to the reconfigurable fabric in the first place. However, there are some additional attack vectors for the resulting device as a whole which are out of the scope here, but need to be considered in any final implementation.

A common target of some attacks on any IC is its test circuitry [210], i.e., DfT infrastructure, such as scan-chains or any other kind of BIST. The test circuitry may reveal secrets directly, if not protected, or may open additional side-channels even if test ports are protected. In addition, DfT may violate security assumptions and weaken the employed security mechanisms.

An attack vector specific to reconfigurable fabric is its bitstream as shown for commercial FPGAs [211]. In particular, bitstream tampering and insertion of Trojans [212] are serious threats. As a countermeasure, it is usually possible to enable encryption and verification of bitstreams loaded onto the FPGA. However, the process of loading an encrypted bitstream can also be attacked. Various SCA attacks show that it is possible to reveal the bitstream encryption key [213]–[215]. Furthermore, the system components involved in loading, decrypting, and verifying bitstream can be attacked, as shown for First-Stage Boot Loader (FSBL) in [216] or executing some malicious code on AMD/Xilinx 7-series SoCs [217]. In addition, adversaries can try to inject faults into the device configuration [218]–[220] or

abuse partial reconfiguration. It might also be possible to execute replay attacks by loading an older version of the bitstream that are not tampered, but with some vulnerabilities that are fixed in the newer versions.

The memory used in the final devices is also a possible attack vector, regardless of its use for configuration or computation. For instance, in [221] the authors demonstrated a critical device vulnerability over the SDRAM interface of SoC FPGA devices.

6.4.6 Open-Source

We want to highlight that our entire methodology is realizable with publicly available tools, as shown by our concept and case study. This is very significant, since this topic becomes easily accessible to the entire scientific community, enabling transparency and research on the subject of secure hardware. In addition, this opens opportunities to implement security features and allows fine-grained adjustments to any step of the development process, such as improving the gadgets with respect to security as stated above, and custom routing algorithms for better balanced wire loads in DRP. Hence, we contribute our implementation by making it publicly available on Github[2].

6.5 Conclusion

In this work, we presented a novel concept for provable secure reconfigurable fabric design based on secure gadgets realizing first-order Boolean Masking and DRP. Furthermore, we implemented a secure fabric as a case study with the assistance of the open-source framework FABulous, emulated a design on a commercial FPGA and conducted experimental SCA evaluations by means of TVLA using 100 million measurements. In particular, we presented how reconfigurable secure gadgets can be implemented based on WDDL_{SC} and LMDPL_{SC} gadgets and composed to construct a fabric. Based on the underlying security models, our proposed approach offers strong protection against SCA attacks and enables the detection of a wide range of fault injection attempts. The employed countermeasures are abstracted at the gate level, and the entire process of generating a configuration bitstream compatible to the fabric for a given HDL design is fully automated by publicly available tools. This would allow any engineer to apply security countermeasures correctly by primary input annotations in the unprotected HDL design and map it to our fabric. Especially, symmetric crypto implementations that can be converted into Boolean-masked circuits efficiently, greatly benefit from this approach. A configured design can be changed in the field by programming a new bitstream, allowing for updates to programmed (security) algorithms. However, integrated countermeasures at the physical level are fixed at the time of device fabrication and cannot be altered post-production.

[2] `https://github.com/ChairImpSec/SAUBER`

Our current approach can be improved from two perspectives. (1) Our case study can detect a reasonably large set of faults, hence should be able to resist DFA attacks. This however does not provide security against some other sophisticated FI attacks like SIFA [84], where having knowledge about effectivity/ineffectivity of the injected faults is enough to conduct the attack. To cope with this, either fault correction facilities should be considered in the design of gadgets, or some independent sensors for different fault injection techniques should be integrated into the fabric. (2) The gadgets employed in our case study can provide security against first-order SCA and DFA attacks, but not necessary against their combination, i.e., when SCA measurements are collected while faults are also injected. Protection against such combined attacks require a fundamentally new design for the gadgets, where both attack vectors are covered from scratch.

7 Conclusion

This thesis systematically advances the field of hardware security by addressing both the identification of novel physical attack vectors and the development of scalable, automated defense methodologies. By challenging established security axioms and providing verifiable mitigation frameworks, the collective contributions of this research bridge the gap between theoretical hardware vulnerabilities and practical, resource-efficient implementations on reconfigurable systems.

The broader implications of this work can be categorized into two primary domains: the reevaluation of foundational hardware threat models and the democratization of secure hardware design.

7.1 Identification of novel physical attack vectors

Our empirical investigations introduce new threat models to the physical security in both test and operational environments:

- **Invalidation of DfT Obscurity:** The findings concerning test-infrastructure-induced side channels demonstrate that restricting access to internal test infrastructure is insufficient for preserving security. By proving that non-sensitive, publicly accessible outputs leak secret-dependent transient behavior data during timing tests, this work mandates a critical revision of DfT threat models. It establishes the necessity for integrating information-leakage evaluations into production databases and safety-critical diagnostic protocols.
- **Side-Channel Leakage in Signal Jitter:** The discovery of JitSCA proves that standard galvanic isolation—traditionally considered a definitive countermeasure against power and electromagnetic side channels—fails to attenuate temporal leakage. By demonstrating that deterministic transient voltage fluctuations propagate as signal jitter across isolated boundaries (e.g., yielding key recovery with as few as 47,000 traces over an isolated HDMI link), this research exposes a pervasive vulnerability in modern physical and link-layer interconnects.

7.2 Secure Hardware Design Automation

A central barrier to the widespread adoption of robust hardware security is the reliance on specialized cryptographic engineering expertise and the high efforts associated with manual countermeasure integration. This thesis resolves these bottlenecks through architecture-aware automation:

- **Efficient Masking for Commercial FPGAs:** The introduction of the AGEMA_FPGA EDA flow demonstrates that formally verified masking paradigms, such as PINI, can be implemented more efficiently on FPGAs. By shifting the application of masking directly to the intrinsic LUT boundaries and isolating linear primitives, the proposed methodology achieves up to a 64% reduction in LUT utilization and a 59% reduction in power consumption compared to conventional ASIC-to-FPGA translations.
- **Verifiable Flexible Hardware Root-of-Trust:** The development of a secure, open-source reconfigurable fabric framework fundamentally solves the secure hardware inflexibility problem. By fully automating the mapping, placement, and routing of unprotected HDL designs onto a platform governed by secure composable gadgets, this work eliminates the need for manual security mapping. The empirical validation—demonstrating an 85% area reduction relative to conventional FPGA baselines and no detectable leakage over 100 million traces—proves that robust, gate-level resilience against SCA and FI attacks can be achieved deterministically.

Ultimately, by shifting the burden of physical security from the application developer to the foundational EDA tooling and platform infrastructure, this thesis establishes a flexible, automated, and formally verifiable hardware root-of-trust suitable for the next generation of critical digital infrastructures.

A Appendix

List of Figures

List of Tables

Abbreviations

AES Advanced Encryption Standard

ARX add–rotate–XOR

ASIC Application Specific Integrated Circuit

ATE Automatic Test Equipment

AWDDL Asynchronous Wave Dynamic Differential Logic

BIST Built-In Self-Test

CDF Cumulative Distribution Function

CGB Configurable Gadget Block

CLB Configurable Logic Block

CMOS Complementary Metal Oxide Semiconductor

CPA Correlation Power Analysis

CPU Central Processing Unit

CUT Circuit Under Test

DfT Design-for-Testability

DDj Data Dependent jitter

DFA Differential Fault Analysis

DRP Dual-Rail Pre-charge Logic

DPA Differential Power Analysis

DRSL Dual-Rail Random Switching Logic

DUT Device Under Test

DVFS Dynamic Voltage and Frequency Scaling

eFPGA Embedded FPGA

EDA Electronic Design Automation

FASM FPGA Assembly

FD Fault Dictionary

FDRE D FF with Clock Enable and Synchronous Reset

FI Fault Injection

FPGA Field Programmable Gate Array

FSA Fault Sensitivity Analysis

FSBL First-Stage Boot Loader

FSM Finite-State Machine

GHPC Generic Hardware Private Circuits

GLM Generic Low-Latency Masking

GPDI General Purpose Differential Interface

GSNI Glitch Strong Non-Interference

HC Higher Criticism

HD Hamming Distance

HDL Hardware Description Language

HPC Hardware Private Circuits

HW Hamming Weight

JitSCA Jitter-based Side-Channel Analysis

JSON JavaScript Object Notation

LFSR Linear Feedback Shift Register

LMDPL LUT-based Masked Dual-Rail with Pre-charge Logic

LMDPL$_{SC}$ Self-Checking LUT-based Masked Dual-Rail with Pre-charge Logic

LUT Look-Up Table

LVDS Low-Voltage Differential Signaling

MDPL Masked Dual-Rail Pre-charge Logic

MISR Multiple Input Signature Register

NIST National Institute of Standards and Technology

NRZI Non-return-to-zero inverted

NRZS Non-return-to-zero space

PDN Power Distribution Network

PINI Probe-Isolating Non-Interference

PLL Phase Locked Loop

PQC Post-quantum cryptography

PRNG Pseudo-Random Number Generator

PSM Parallel Switch Matrix

RNG Random Number Generator

RO Ring Oscillator

RTL Register Transfer Level

SABL Sense Amplifier Based Logic

SCA Side-Channel Analysis

SDD Small Delay Defect

SIFA Statistical Ineffective Fault Analysis

IFA Ineffective Fault Analysis

SNI Strong Non-Interference

SNR Signal to Noise Ratio

SPA Simple Power Analysis

TAP Test Access Point

TDC Time-to-Digital Converter

TPM Trusted Platform Module

TRNG True Random Number Generator

TVLA Test Vector Leakage Assessment

SoC System on Chip

WDDL Wave Dynamic Differential Logic

WDDL$_{SC}$ Self-Checking Wave Dynamic Differential Logic

GE Gate Equivalent

HSM Hardware Security Module

ECU Electronic Control Unit

CAN Controller Area Network

POS Point-of-Sale

PIN Personal Identification Number

Bibliography

[1] S. Meschkov, D. R. E. Gnad, J. Krautter, and M. B. Tahoori, "New approaches of side-channel attacks based on chip testing methods", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2022.

[2] K. Schoos, S. Meschkov, M. B. Tahoori, and D. R. Gnad, "Jitsca: Jitter-based side-channel analysis in picoscale resolution", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 3, pp. 294–320, 2023.

[3] S. Meschkov, D. Lammers, M. B. Tahoori, and A. Moradi, "Design and implementation of a physically secure open-source fpga and toolchain", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2025, no. 3, pp. 542–582, 2025.

[4] S. Meschkov, D. R. Gnad, J. Krautter, and M. B. Tahoori, "Is your secure test infrastructure secure enough?: Attacks based on delay test patterns using transient behavior analysis", in *2021 IEEE International Test Conference (ITC)*, IEEE, 2021, pp. 334–338.

[5] N. Muller, S. Meschkov, D. R. Gnad, M. B. Tahoori, and A. Moradi, "Automated masking of fpga-mapped designs", in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, IEEE, 2023, pp. 79–85.

[6] H. Xu, S. Meschkov, V. Meyers, and M. Tahoori, "Pwnn: Power-wasting neural network as remote fault injector", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2026, no. 1, pp. 448–471, 2026.

[7] B. Sapui, J. Krautter, M. Mayahinia, *et al.*, "Power side-channel attacks and countermeasures on computation-in-memory architectures and technologies", in *2023 IEEE European Test Symposium (ETS)*, IEEE, 2023, pp. 1–6.

[8] B. Sapui, S. Meschkov, and M. B. Tahoori, "Side-channel attack with fault analysis on memristor-based computation-in-memory", in *IEEE 30th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, 2024.

[9] S. M. Ghasemi, S. Meschkov, J. Krautter, D. R. Gnad, and M. B. Tahoori, "Enabling in-field parametric testing for risc-v cores", in *2023 IEEE International Test Conference (ITC)*, IEEE, 2023, pp. 367–376.

[10] S. M. Ghasemi, S. Meschkov, J. Krautter, D. R. Gnad, and M. B. Tahoori, "Slm isa and hardware extensions for risc-v processors", in *2023 IEEE 29th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, IEEE, 2023, pp. 1–5.

[11] S. M. Ghasemi, J. Krautter, T. Gheshlaghi, S. Meschkov, D. R. Gnad, and M. B. Tahoori, "Degradation monitoring through software-controlled on-chip sensors for risc-v", in *2024 IEEE European Test Symposium (ETS)*, IEEE, 2024, pp. 1–6.

[12] S. M. Ghasemi, S. Meschkov, J. Krautter, D. R. Gnad, and M. B. Tahoori, "In-field detection of small delay defects and runtime degradation using on-chip sensors", in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2024, pp. 1–2.

[13] A. M. Research, *Hardware encryption market statistics, 2032*, https://www.alliedmarketresearch.com/hardware-encryption-market, Accessed: 01 March 2025.

[14] A. Kia, A. W. Storey, and M. Imtiaz, "Advanced hardware security on embedded processors: A 2026 systematic review", *Electronics*, vol. 15, no. 5, 2026, ISSN: 2079-9292. DOI: 10.3390/electronics15051135. [Online]. Available: https://www.mdpi.com/2079-9292/15/5/1135.

[15] IBM Security and Ponemon Institute, "Cost of a data breach report 2025", IBM Corporation, Annual Research Report, 2025, Accessed: March 16, 2026. [Online]. Available: https://www.ibm.com/reports/data-breach.

[16] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems", in *CRYPTO '96 - Annual International Cryptology Conference*. Berlin, Heidelberg: Springer, 1996, pp. 104–113, ISBN: 978-3-540-68697-2. DOI: 10.1007/3-540-68697-5_9.

[17] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis", in *Advances in Cryptology*, Springer Berlin Heidelberg, 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25.

[18] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side—channel (s)", in *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*, Springer, 2003, pp. 29–45.

[19] 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. DOI: 10.1007/3-540-69053-0_4.

[20] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems", in *Advances in Cryptology — CRYPTO '97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings*, B. S. Kaliski, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 513–525, ISBN: 978-3-540-69528-8. DOI: 10.1007/BFb0052259.

[21] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis", in *Cryptographic Hardware and Embedded Systems, CHES 2010*, 2010.

[22] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model", *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 16–29, 2004. DOI: 10.1007/978-3-540-28632-5_2.

[23] G. Becker, J. Cooper, E. DeMulder, *et al.*, "Test vector leakage assessment (tvla) methodology in practice", in *International Cryptographic Module Conference*, vol. 1001, 2013, p. 13.

[24] D. Knichel, A. Moradi, N. Müller, and P. Sasdrich, "Automated generation of masked hardware", *Cryptology ePrint Archive*, 2021.

[25] D. Knichel, P. Sasdrich, and A. Moradi, "Generic hardware private circuits: Towards automated generation of composable secure gadgets", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 323–344, 2022.

[26] G. Cassiers and F. Standaert, "Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference", *IEEE TIFS*, vol. 15, pp. 2542–2555, 2020.

[27] M. Stojilović, K. Rasmussen, F. Regazzoni, M. B. Tahoori, and R. Tessier, "A visionary look at the security of reconfigurable cloud computing", *Proceedings of the IEEE*, vol. 111, no. 12, pp. 1548–1571, 2023. DOI: 10.1109/JPROC.2023.3330729.

[28] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on fpgas", in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2018, pp. 1111–1116.

[29] J. Krautter, D. R. 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*, pp. 44–68, 2018.

[30] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks", in *Symposium on Security and Privacy (SP)*, IEEE, May 2018, pp. 805–820. DOI: 10.1109/SP.2018.00049.

[31] 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 (TCHES)*, vol. 2019, no. 3, pp. 305–339, May 2019. DOI: 10.13154/tches.v2019.i3.305-339.

[32] E. Amouri, Z. Marrakchi, and H. Mehrez, "Controlled placement and routing techniques to improve timing balance of wddl designs in mesh-based fpga", in *2010 IEEE Asia Pacific Conference on Circuits and Systems*, IEEE, 2010, pp. 296–299.

[33] E. Amouri, H. Mehrez, and Z. Marrakchi, "Impact of dual placement and routing on wddl netlist security in fpga", *International Journal of Reconfigurable Computing*, vol. 2013, no. 1, p. 802 436, 2013.

[34] E. Amouri, S. Bhasin, Y. Mathieu, T. Graba, J.-L. Danger, and H. Mehrez, "Balancing wddl dual-rail logic in a tree-based fpga to enhance physical security", in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, IEEE, 2014, pp. 1–4.

[35] A. Moradi and V. Immler, "Early propagation and imbalanced routing, how to diminish in fpgas", in *Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings 16*, Springer, 2014, pp. 598–615.

[36] S. Chaudhuri, S. Guilley, P. Hoogvorst, *et al.*, "Physical design of fpga interconnect to prevent information leakage", in *International Workshop on Applied Reconfigurable Computing*, Springer, 2008, pp. 87–98.

[37] T. Beyrouthy, A. Razafindraibe, L. Fesquet, *et al.*, "A novel asynchronous e-fpga architecture for security applications", in *2007 International Conference on Field-Programmable Technology*, IEEE, 2007, pp. 369–372.

[38] A. Mokari, B. Ghavami, and H. Pedram, "Scar-fpga: A novel side-channel attack resistant fpga", in *2009 5th Southern Conference on Programmable Logic (SPL)*, IEEE, 2009, pp. 177–182.

[39] D. Koch, N. Dao, B. Healy, J. Yu, and A. Attwood, "Fabulous: An embedded fpga framework", in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2021, pp. 45–56.

[40] C. Wolf, J. Glaser, and J. Kepler, "Yosys-a free verilog synthesis suite", 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:202611483.

[41] D. Shah, E. Hung, C. Wolf, S. Bazanski, D. Gisselquist, and M. Milanovic, "Yosys+nextpnr: An open source framework from verilog to bitstream for commercial fpgas", *CoRR*, vol. abs/1903.10407, 2019. arXiv: 1903.10407. [Online]. Available: http://arxiv.org/abs/1903.10407.

[42] K. Dang Pham, E. Horta, and D. Koch, "Bitman: A tool and api for fpga bitstream manipulations", in *Design, Automation, and Test in Europe Conference and Exhibition (DATE), 2017*, 2017, pp. 894–897. DOI: 10.23919/DATE.2017.7927114.

[43] *BitMan GitHub*, https://github.com/khoapham/bitman, Accessed: 2024-01-12.

[44] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure dpa resistant asic or fpga implementation", in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, IEEE, vol. 1, 2004, pp. 246–251.

[45] P. Sasdrich, B. Bilgin, M. Hutter, and M. E. Marson, "Low-latency hardware masking with application to aes", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 300–326, 2020.

[46] Lam, "A theory of totally self-checking system design", *IEEE transactions on computers*, vol. 100, no. 9, pp. 831–844, 1983.

[47] C. O'Flynn and A. Dewar, "On-device power analysis across hardware security domains", *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, pp. 126–153, 2019.

[48] 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. DOI: 10.1145/3240765.3240841.

[49] T. Korak and T. Plos, "Applying remote side-channel analysis attacks on a security-enabled nfc tag", in *Cryptographers' Track at the RSA Conference*, Springer, 2013, pp. 207–222.

[50] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming channels: When electromagnetic side channels meet radio transceivers", in *Conference on Computer and Communications Security (CCS)*, Toronto, Canada: ACM, Oct. 2018.

[51] Z. Zhan, Z. Zhang, S. Liang, F. Yao, and X. Koutsoukos, "Graphics peeping unit: Exploiting em side-channel information of gpus to eavesdrop on your neighbors", in *Symposium on Security and Privacy (SP)*, IEEE, 2022.

[52] K. M. Zick and J. P. Hayes, "Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems", *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 5, no. 1, pp. 1–26, 2012.

[53] D. Spielmann, O. Glamočanin, and M. Stojilović, "Rds: Fpga routing delay sensors for effective remote power analysis attacks", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 2, pp. 543–567, 2023.

[54] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked aes hardware implementations", in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2005, pp. 157–171.

[55] J.-S. Coron, C. Giraud, E. Prouff, S. Renner, M. Rivain, and P. K. Vadnala, "Conversion of security proofs from one leakage model to another: A new issue", in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, Springer, 2012, pp. 69–81.

[56] T. De Cnudde, B. Bilgin, B. Gierlichs, V. Nikov, S. Nikova, and V. Rijmen, "Does coupling affect the security of masked implementations?", in *Constructive Side-Channel Analysis and Secure Design: 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers 8*, Springer, 2017, pp. 1–18.

[57] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolić, *Digital integrated circuits: a design perspective*. Pearson Education, Incorporated., 2003.

[58] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks", in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2002.

[59] T. Schneider and A. Moradi, "Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations", in *CHES 2015*, ser. LNCS, vol. 9293, Springer, 2015, pp. 495–513.

[60] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards", in *Proceedings of the 28th European solid-state circuits conference*, IEEE, 2002, pp. 403–406.

[61] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints", in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2005, pp. 172–186.

[62] Z. Chen and Y. Zhou, "Dual-rail random switching logic: A countermeasure to reduce side channel leakage", in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2006, pp. 242–254.

[63] D. Suzuki and M. Saeki, "Security evaluation of dpa countermeasures using dual-rail pre-charge logic style", in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2006, pp. 255–269.

[64] K. Tiri and I. Verbauwhede, "Place and route for secure standard cell design", in *Smart Card Research and Advanced Applications VI: IFIP 18th World Computer Congress TC8/WG8. 8 & TC11/WG11. 2 Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS) 22–27 August 2004 Toulouse, France*, Springer, 2004, pp. 143–158.

[65] S. Bhasin, S. Guilley, F. Flament, N. Selmane, and J. Danger, "Countering early evaluation: An approach towards robust dual-rail precharge logic", in *WESS*, ACM, 2010, p. 6.

[66] A. Moradi and A. Wild, "Assessment of hiding the higher-order leakages in hardware: What are the achievements versus overheads?", in *Cryptographic Hardware and Embedded Systems–CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17*, Springer, 2015, pp. 453–474.

[67] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks", in *Advances in Cryptology*, M. Wiener, Ed., Springer Berlin Heidelberg, 1999, pp. 398–412, ISBN: 978-3-540-48405-9.

[68] A. Shamir, "How to share a secret", *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[69] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks", in *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, Springer, 2003, pp. 463–481.

[70] S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F. Standaert, "Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model", *IACR TCHES*, vol. 2018, no. 3, pp. 89–120, 2018.

[71] G. Cassiers, B. Grégoire, I. Levi, and F.-X. Standaert, "Hardware private circuits: From trivial composition to full verification", *IEEE Transactions on Computers*, vol. 70, no. 10, pp. 1677–1690, 2020.

[72] D. K. S. V., S. Dhooghe, J. Balasch, B. Gierlichs, and I. Verbauwhede, "Time sharing - A novel approach to low-latency masking", *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2024, no. 3, pp. 249–272, 2024. DOI: 10.46586/TCHES.V2024.I3.249-272. [Online]. Available: https://doi.org/10.46586/tches.v2024.i3.249-272.

[73] G. Barthe, S. Belaïd, F. Dupressoir, *et al.*, "Strong Non-Interference and Type-Directed Higher-Order Masking", in *CCS 2016*, ACM, 2016, pp. 116–129.

[74] L. De Meyer, B. Bilgin, and O. Reparaz, "Consolidating security notions in hardware masking", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 119–147, 2019.

[75] D. Knichel, P. Sasdrich, and A. Moradi, "Generic Hardware Private Circuits Towards Automated Generation of Composable Secure Gadgets", *IACR TCHES*, vol. 2022, no. 1, pp. 323–344, 2022.

[76] A. J. Leiserson, M. E. Marson, and M. A. Wachs, "Gate-level masking under a path-based leakage metric", in *Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings 16*, Springer, 2014, pp. 580–597.

[77] N. Müller, D. Lammers, and A. Moradi, "A deep analysis of two glitch-free hardware masking schemes SESYM and LMDPL", *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2024, no. 3, pp. 76–98, 2024. DOI: 10.46586/TCHES.V2024.I3.76-98. [Online]. Available: https://doi.org/10.46586/tches.v2024.i3.76-98.

[78] D. Knichel and A. Moradi, "Low-latency hardware private circuits", in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1799–1812.

[79] R. Anderson and M. Kuhn, "Tamper resistance-a cautionary note", in *Proceedings of the second Usenix workshop on electronic commerce*, vol. 2, 1996, pp. 1–11.

[80] N. T. Courtois, K. Jackson, and D. Ware, "Fault-algebraic attacks on inner rounds of des", in *e-Smart'10 Proceedings: The Future of Digital Security Technologies*, Strategies Telecom and Multimedia, 2010.

[81] J. Blömer and V. Krummel, "Fault based collision attacks on aes", in *International Workshop on Fault Diagnosis and Tolerance in Cryptography*, Springer, 2006, pp. 106–120.

[82] R. Schilling, M. Werner, and S. Mangard, "Securing conditional branches in the presence of fault attacks", in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2018, pp. 1586–1591.

[83] C. Clavier, "Secret external encodings do not prevent transient fault analysis", in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2007, pp. 181–194.

[84] C. Dobraunig, M. Eichlseder, T. Korak, S. Mangard, F. Mendel, and R. Primas, "Sifa: Exploiting ineffective fault inductions on symmetric cryptography", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, Aug. 2018. DOI: 10.13154/tches.v2018.i3.547-572.

[85] M. Gross, J. Krautter, D. Gnad, M. Gruber, G. Sigl, and M. Tahoori, "Fpganeedle: Precise remote fault attacks from fpga to cpu", in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 2023, pp. 358–364.

[86] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis", in *Cryptographic Hardware and Embedded Systems, CHES 2010*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ISBN: 978-3-642-15031-9.

[87] F. Melzani and A. Palomba, "Enhancing fault sensitivity analysis through templates", in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013. DOI: 10.1109/HST.2013.6581560.

[88] Q. Wang, A. Wang, G. Qu, and G. Zhang, "New methods of template attack based on fault sensitivity analysis", *IEEE Transactions on Multi-Scale Computing Systems*, 2017. DOI: 10.1109/TMSCS.2016.2643638.

[89] S. Endo, Y. Li, N. Homma, *et al.*, "A silicon-level countermeasure against fault sensitivity analysis and its evaluation", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 8, pp. 1429–1438, 2014.

[90] S. Ghosh, D. Saha, A. Sengupta, and D. R. Chowdhury, "Preventing fault attacks using fault randomisation with a case study on aes", *International Journal of Applied Cryptography*, vol. 3, no. 3, pp. 225–235, 2017.

[91] A. R. Shahmirzadi, S. Rasoolzadeh, and A. Moradi, "Impeccable circuits ii", in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, IEEE, 2020, pp. 1–6.

[92] N. Moro, K. Heydemann, E. Encrenaz, and B. Robisson, "Formal verification of a software countermeasure against instruction skip attacks", *Journal of Cryptographic Engineering*, vol. 4, no. 3, pp. 145–156, 2014.

[93] M. Medwed, F.-X. Standaert, J. Großschädl, and F. Regazzoni, "Fresh re-keying: Security against side-channel and fault attacks for low-cost devices", in *International Conference on Cryptology in Africa*, Springer, 2010, pp. 279–296.

[94] A. Baksi, S. Bhasin, J. Breier, M. Khairallah, and T. Peyrin, "Protecting block ciphers against differential fault attacks without re-keying (extended version)", *Cryptology ePrint Archive*, 2018.

[95] A. Baksi, "Classical and physical security of symmetric key cryptographic algorithms", in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, IEEE, 2021, pp. 1–2.

[96] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard", in *International Test Conference*, IEEE, 2004.

[97] R. Nara, K. Satoh, M. Yanagisawa, T. Ohtsuki, and N. Togawa, "Scan-based side-channel attack against rsa cryptosystems using scan signatures", *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 2010.

[98] X. Li, W. Li, J. Ye, H. Li, and Y. Hu, "Scan chain based attacks and countermeasures: A survey", *IEEE Access*, 2019.

[99] A. Cui, Y. Luo, H. Li, and G. Qu, "Why current secure scan designs fail and how to fix them?", *Integration*, 2017, ISSN: 0167-9260. DOI: https://doi.org/10.1016/j.vlsi.2016.10.011.

[100] J. Da Rolt, G. Di Natale, M. Flottes, and B. Rouzeyre, "New security threats against chips containing scan chain structures", in *International Symposium on Hardware-Oriented Security and Trust*, IEEE, 2011. DOI: 10.1109/HST.2011.5955005.

[101] J. Da Rolt, A. Das, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede, "Test versus security: Past and present", *IEEE Transactions on Emerging topics in Computing*, 2014.

[102] E. DeBusschere and M. McCambridge, "Modern game console exploitation", *Tech. Report, Dept. of Comp. Sci., University of Arizona*, 2012.

[103] Anonymous Hacker, *Xbox 360 hypervisor privilege escalation vulnerability*, Feb. 2007. [Online]. Available: `http://securityvulns.com/Qdocument211.html`.

[104] R.-P. Weinmann, "Baseband attacks: Remote exploitation of memory corruptions in cellular protocol stacks.", in *WOOT*, 2012.

[105] B. Yang, K. Wu, and R. Karri, "Secure scan: A design-for-test architecture for crypto chips", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2006. DOI: `10.1109/TCAD.2005.862745`.

[106] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, "Securing designs against scan-based side-channel attacks", *IEEE Transactions on Dependable and Secure Computing*, 2007. DOI: `10.1109/TDSC.2007.70215`.

[107] K. Rosenfeld and R. Karri, "Attacks and defenses for jtag", *IEEE Design & Test of Computers*, 2010.

[108] M. Da Silva, M.-L. Flottes, G. Di Natale, B. Rouzeyre, P. Prinetto, and M. Restifo, "Scan chain encryption for the test, diagnosis and debug of secure circuits", in *European Test Symposium (ETS)*, IEEE, 2017.

[109] M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Preventing scan attacks on secure circuits through scan chain encryption", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[110] D. Vaghani, S. Ahlawat, J. Tudu, M. Fujita, and V. Singh, "On securing scan design through test vector encryption", in *International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2018.

[111] E. Valea, M. Da Silva, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Encryption-based secure jtag", in *International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, IEEE, 2019.

[112] ARM, *Trustzone*, 2012. [Online]. Available: `http://www.atmel.com/Images/doc8558.pdf`.

[113] S. Burton, J. Likkei, P. Vembar, and M. Wolf, "Automotive functional safety= safety+ security", in *Proceedings of the First International Conference on Security of Internet of Things*, 2012, pp. 150–159.

[114] K. S. Kim, S. Mitra, and P. G. Ryan, "Delay defect characteristics and testing strategies", *IEEE Design & Test of Computers*, 2003.

[115] R. Franch, P. Restle, N. James, *et al.*, "On-chip timing uncertainty measurements on ibm microprocessors", in *International Test Conference*, IEEE, IEEE, Oct. 2007, pp. 1–7. DOI: `10.1109/TEST.2007.4437560`.

[116] J. Zeng, M. S. Abadir, G. Vandling, L.-C. Wang, S. Karako, and J. A. Abraham, "On correlating structural tests with functional tests for speed binning of high performance design", in *Fifth International Workshop on Microprocessor Test and Verification (MTV'04)*, IEEE, 2004.

[117] J. Chen, L.-C. Wang, P.-H. Chang, J. Zeng, S. Yu, and M. Mateja, "Data learning techniques and methodology for fmax prediction", in *International Test Conference*, IEEE, 2009.

[118] H. Yan and A. D. Singh, "A new delay test based on delay defect detection within slack intervals (ddsi)", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2006. DOI: 10.1109/TVLSI.2006.886415.

[119] C. Han and A. D. Singh, "Improving cmos open defect coverage using hazard activated tests", in *VLSI Test Symposium (VTS)*, 2014. DOI: 10.1109/VTS.2014.6818740.

[120] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in ic manufacturing and test", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2018. DOI: 10.1109/TVLSI.2018.2797019.

[121] *Layered Security for Your Next SoC*, https://www.arm.com/products/silicon-ip-security, Accessed: 2024-01-12.

[122] K. Ryu, D.-H. Jung, and S.-O. Jung, "All-digital process-variation-calibrated timing generator for ate with 1.95-ps resolution and a maximum 1.2-ghz test rate", in *Proceedings of the ESSCIRC*, Sep. 2013, ISBN: 978-1-4799-0643-7. DOI: 10.1109/ESSCIRC.2013.6649067.

[123] *Eva100 powerful support for characteristics evaluation, functional evaluation, and production*, 2020. [Online]. Available: https://www3.advantest.com/documents/11348/321634/pdf_EVA100_catalog_en.

[124] S. Hellebrand, T. Indlekofer, M. Kampmann, M. A. Kochte, C. Liu, and H.-J. Wunderlich, "Fast-bist: Faster-than-at-speed bist targeting hidden delay defects", in *International Test Conference*, IEEE, 2014.

[125] R. Tayade and J. A. Abraham, "On-chip programmable capture for accurate path delay test and characterization", in *International Test Conference*, IEEE, 2008.

[126] X. Wang, M. Tehranipoor, and R. Datta, "Path-ro: A novel on-chip critical path delay measurement under process variations", in *IEEE/ACM International Conference on Computer-Aided Design*, IEEE, 2008.

[127] R. Velegalati, K. Shah, and J.-P. Kaps, "Glitch detection in hardware implementations on fpgas using delay based sampling techniques", in *Euromicro Conference on Digital System Design*, IEEE, 2013.

[128] Mentor Graphics, *High quality test solutions for secure applications – silicon test and yield analysis whitepaper*, Apr. 2010.

[129] G. Tshagharyan, G. Harutyunyan, S. Shoukourian, and Y. Zorian, "Securing test infrastructure of system-on-chips", in *East-West Design & Test Symposium (EWDTS)*, IEEE, 2016.

[130] A. Barenghi, L. Breveglieri, A. Palomba, and G. Pelosi, "Fault sensitivity analysis at design time", in *Trusted Computing for Embedded Systems*. Cham: Springer International Publishing, 2015, pp. 175–186, ISBN: 978-3-319-09420-5. DOI: 10.1007/978-3-319-09420-5_9. [Online]. Available: https://doi.org/10.1007/978-3-319-09420-5_9.

[131] M. Abramovici and M. A. Breuer, "Fault diagnosis based on effect-cause analysis: An introduction", in *Proceedings of the 17th Design Automation Conference*, 1980.

[132] E. J. McCluskey, "Built-in self-test techniques", *IEEE Design & Test of Computers*, vol. 2, no. 2, pp. 21–28, 1985.

[133] A. Stroele, "Test response compaction using arithmetic functions", in *Proceedings of 14th VLSI Test Symposium*, 1996, pp. 380–386. DOI: 10.1109/VTEST.1996.510882.

[134] F. Elguibaly and M. El-Kharashi, "Multiple-input signature registers: An improved design", in *1997 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, PACRIM. 10 Years Networking the Pacific Rim, 1987-1997*, vol. 2, 1997, 519–522 vol.2. DOI: 10.1109/PACRIM.1997.620315.

[135] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "An experimental evaluation and analysis of transient voltage fluctuations in FPGAs", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 10, pp. 1817–1830, Oct. 2018, ISSN: 1063-8210. DOI: 10.1109/TVLSI.2018.2848460.

[136] J. Krautter, D. Gnad, and M. Tahoori, "CPAmap: on the complexity of secure fpga virtualization, multi-tenancy, and physical design", *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, 2020.

[137] D. R. E. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on fpgas using valid bitstreams", in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–7. DOI: 10.23919/FPL.2017.8056840.

[138] H. Ma, J. He, Y. Liu, *et al.*, "On-chip trust evaluation utilizing tdc-based parameter-adjustable security primitive", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 10, pp. 1985–1994, 2021. DOI: 10.1109/TCAD.2020.3035346.

[139] A. Moradi, O. Mischke, and C. Paar, "One attack to rule them all: Collision timing attack versus 42 aes asic cores", *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1786–1798, 2013. DOI: 10.1109/TC.2012.154.

[140] B. B. Brumley and N. Tuveri, "Remote timing attacks are still practical", in *European Symposium on Research in Computer Security*, Springer, 2011, pp. 355–371.

[141] D. Moghimi, B. Sunar, T. Eisenbarth, and N. Heninger, "TPM-FAIL: TPM meets timing and lattice attacks", in *USENIX Security Symposium*, 2020, pp. 2057–2073.

[142] P. Kocher, D. Genkin, D. Gruss, *et al.*, "Spectre attacks: Exploiting speculative execution", *ArXiv e-prints*, Jan. 2018. arXiv: 1801.01203 [cs.CR].

[143] M. Yan, R. Sprabery, B. Gopireddy, C. Fletcher, R. Campbell, and J. Torrellas, "Attack directories, not caches: Side channel attacks in a non-inclusive world", in *Symposium on Security and Privacy (SP)*, IEEE, 2019, pp. 888–904.

[144] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware", *Journal of Cryptographic Engineering*, vol. 8, no. 1, pp. 1–27, 2018.

[145] J. Szefer, "Survey of microarchitectural side and covert channels, attacks, and defenses", *Journal of Hardware and Systems Security*, vol. 3, no. 3, pp. 219–234, 2019.

[146] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs", in *Design, Automation & Test in Europe (DATE)*, Mar. 2018.

[147] Y. Wang, R. Paccagnella, E. T. He, H. Shacham, C. W. Fletcher, and D. Kohlbrenner, "Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86", in *USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 679–697.

[148] J. Gravellier, J.-M. Dutertre, Y. Teglia, and P. L. Moundi, "Sideline: How delay-lines (may) leak secrets from your soc", in *Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*, Springer, 2021, pp. 3–30.

[149] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "C^{3}apsule: Cross-fpga covert-channel attacks through power supply unit leakage", in *Symposium on Security and Privacy (SP)*, IEEE, 2020, pp. 1728–1741.

[150] 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.

[151] M. Wang, S. Xie, P. N. Li, *et al.*, "Galvanically isolated, power and electromagnetic side-channel attack resilient secure aes core with integrated charge pump based power management", in *Custom Integrated Circuits Conference (CICC)*, IEEE, 2021, pp. 1–2.

[152] S. Ohira, A. K. Desta, I. Arai, and K. Fujikawa, "PLI-TDC: Super fine delay-time based physical-layer identification with time-to-digital converter for in-vehicle networks", in *Asia Conference on Computer and Communications Security (AsiaCCS)*, ACM, 2021, pp. 176–186.

[153] S. Moini, A. Deric, X. Li, *et al.*, "Voltage sensor implementations for remote power attacks on fpgas", *Transactions on Reconfigurable Technology and Systems (TRETS)*, 2022.

[154] Fibersystem, *Ethernet isolator*, Jan. 2023. [Online]. Available: `https://www.fibersystem.com/product-category/ethernet-isolator/`.

[155] Industrial Control Systems Cyber Emergency Response Team, *Recommended practice: Improving industrial control system cybersecurity with defense-in-depth strategies*, U.S. Department of Homeland Security, Sep. 2016.

[156] A. Moradi, M. Kasper, and C. Paar, "Black-box side-channel attacks highlight the importance of countermeasures", in *Cryptographers' Track at the RSA Conference (CT-RSA)* (Lecture Notes in Computer Science), O. Dunkelman, Ed., Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, ch. Topics in Cryptology CT-RSA, pp. 1–18, ISBN: 978-3-642-27954-6. DOI: 10.1007/978-3-642-27954-6_1. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-27954-6_1.

[157] M. Lipp, A. Kogler, D. Oswald, *et al.*, "PLATYPUS: Software-based power side-channel attacks on x86", in *Symposium on Security and Privacy (SP)*, IEEE, 2021, pp. 355–371.

[158] D. Genkin, N. Nissan, R. Schuster, and E. Tromer, "Lend me your ear: Passive remote physical side channels on PCs", in *USENIX Security Symposium (USENIX Security)*, 2022, pp. 4437–4454.

[159] M. Dworkin, E. Barker, J. Nechvatal, *et al.*, *Advanced encryption standard (aes)*, en, 2001. DOI: https://doi.org/10.6028/NIST.FIPS.197.

[160] 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.

[161] A. A. Ding, L. Zhang, F. Durvaux, F.-X. Standaert, and Y. Fei, "Towards sound and optimal leakage detection procedure", in *Smart Card Research and Advanced Applications*, T. Eisenbarth and Y. Teglia, Eds., Cham: Springer International Publishing, 2018, pp. 105–122, ISBN: 978-3-319-75208-2.

[162] J. Kalisz, "Review of methods for time interval measurements with picosecond resolution", *Metrologia*, vol. 41, no. 1, p. 17, 2003.

[163] C. Lamech, J. Aarestad, J. Plusquellic, R. Rad, and K. Agarwal, "REBEL and TDC: Two embedded test structures for on-chip measurements of within-die path delay variations", in *International Conference on Computer-Aided Design (ICCAD)*, IEEE/ACM, 2011, pp. 170–177. DOI: 10.1109/ICCAD.2011.6105322.

[164] Teledyne LeCroy, *Understanding jitter calculations: Why dj can be less than ddj (or pj)*, https://teledynelecroy.com/doc/understanding-dj-ddj-pj-jitter-calculations, Jul. 2014. [Online]. Available: https://teledynelecroy.com/doc/understanding-dj-ddj-pj-jitter-calculations.

[165] Analog Devices, *Galvanic Isolation of the HDMI 1.3a Protocol Using iCoupler® Isolation Technology*, 2013. [Online]. Available: https://www.analog.com/en/design-center/reference-designs/circuits-from-the-lab/CN0422.html.

[166] E. Salem, H. L. Abdelhalim Zekry, and R. Tawfik, "Fpga implementation of 1000base-x ethernet physical layer core", *International Journal of Engineering & Technology*, vol. 7, no. 4, pp. 2106–2112, 2018.

[167] M. Guri, B. Zadov, D. Bykhovsky, and Y. Elovici, "Powerhammer: Exfiltrating data from air-gapped computers through power lines", *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1879–1890, 2019.

[168] Weisstechnik, *EMC test chambers LabEvent*, 2021. [Online]. Available: https://backend.weiss-technik.com/webapp/weisstechnik/multimedia-center/brochures/2021/Weiss-Technik-LabEvent-EMV-EN-1.pdf.

[169] Genua. Highly Secure Industrial Monitoring, *An industrial data diode for especially critical plants and processes*, Apr. 2023. [Online]. Available: https://www.genua.eu/it-security-solutions/data-diode-cyber-diode.

[170] T. Güneysu, "FPGAs in Cryptography", in *Encyclopedia of Cryptography and Security, 2nd Ed*, Springer, 2011, pp. 499–501.

[171] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", in *CRYPTO 1996*, ser. LNCS, vol. 1109, Springer, 1996, pp. 104–113.

[172] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", in *CRYPTO 1999*, ser. LNCS, vol. 1666, Springer, 1999, pp. 388–397.

[173] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic Analysis: Concrete Results", in *CHES 2001*, ser. LNCS, vol. 2162, Springer, 2001, pp. 251–261.

[174] M. Hutter and J. Schmidt, "The Temperature Side Channel and Heating Fault Attacks", in *CARDIS 2013*, ser. LNCS, vol. 8419, Springer, 2013, pp. 219–235.

[175] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks", in *CRYPTO 1999*, ser. LNCS, vol. 1666, Springer, 1999, pp. 398–412.

[176] Y. Ishai, A. Sahai, and D. A. Wagner, "Private Circuits: Securing Hardware against Probing Attacks", in *CRYPTO 2003*, ser. LNCS, vol. 2729, Springer, 2003, pp. 463–481.

[177] S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F. Standaert, "Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model", *IACR TCHES*, vol. 2018, no. 3, pp. 89–120, 2018.

[178] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold Implementations Against Side-Channel Attacks and Glitches", in *ICICS 2006*, ser. LNCS, vol. 4307, Springer, 2006, pp. 529–545.

[179] T. Moos, A. Moradi, T. Schneider, and F. Standaert, "Glitch-Resistant Masking Revisited or Why Proofs in the Robust Probing Model are Needed", *IACR TCHES*, vol. 2019, no. 2, pp. 256–292, 2019.

[180] G. Cassiers, B. Grégoire, I. Levi, and F. Standaert, "Hardware Private Circuits: From Trivial Composition to Full Verification", *IEEE Trans. Comp.*, vol. 70, no. 10, pp. 1677–1690, 2021.

[181] D. Knichel and A. Moradi, "Low-Latency Hardware Private Circuits", in *CCS 2022*, ACM, 2022, pp. 1799–1812.

[182] D. Knichel, A. Moradi, N. Müller, and P. Sasdrich, "Automated Generation of Masked Hardware", *IACR TCHES*, vol. 2022, no. 1, pp. 589–629, 2022.

[DC] S. Inc., *Design compiler graphical*, https://www.synopsys.com/.

[183] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard* (Information Security and Cryptography). Springer, 2002.

[184] C. Beierle, G. Leander, A. Moradi, and S. Rasoolzadeh, "CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks", *IACR Trans. Symmetric Cryptol.*, vol. 2019, no. 1, pp. 5–45, 2019.

[185] S. Banik, A. Bogdanov, T. Isobe, *et al.*, "Midori: A Block Cipher for Low Energy", in *ASIACRYPT*, ser. LNCS, vol. 9453, Springer, 2015, pp. 411–436.

[186] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw, "The LED Block Cipher", in *CHES*, ser. LNCS, vol. 6917, Springer, 2011, pp. 326–341.

[187] C. Beierle, J. Jean, S. Kölbl, *et al.*, "The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS", in *CRYPTO*, ser. LNCS, vol. 9815, Springer, 2016, pp. 123–153.

[188] L. D. Meyer, A. Moradi, and F. Wegener, "Spin Me Right Round Rotational Symmetry for FPGA-Specific AES", *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2018, no. 3, pp. 596–626, 2018.

[189] SAKURA, *Side-channel Attack User Reference Architecture*, `http://satoh.cs.uec.ac.jp/SAKURA/index.html`, 2016.

[190] B. L. Synthesis and V. Group, *Abc: A system for sequential synthesis and verification*, `https://people.eecs.berkeley.edu/~alanmi/abc/`, Accessed: 2025-01-13, 2025.

[191] V.-R. (Project, *Vpr: Versatile place and route*, `https://docs.verilogtorouting.org/en/latest/vpr/`, Accessed: 2025-01-13, 2025.

[192] A. Rukhin, J. Soto, J. Nechvatal, *et al.*, *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. US Department of Commerce, Technology Administration, National Institute of ..., 2001, vol. 22.

[193] P. Sasdrich, B. Bilgin, M. Hutter, and M. E. Marson, "Low-latency hardware masking with application to AES", *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 2, pp. 300–326, 2020.

[194] G. Cassiers, L. Masure, C. Momin, T. Moos, A. Moradi, and F.-X. Standaert, "Randomness generation for secure hardware masking-unrolled trivium to the rescue", *Cryptology ePrint Archive*, 2023.

[195] H. Groß, R. Iusupov, and R. Bloem, "Generic low-latency masking in hardware", *IACR transactions on cryptographic hardware and embedded systems*, pp. 1–21, 2018.

[196] J. Daemen and V. Rijmen, "Aes proposal: Rijndael", 1999.

[197] S. Banik, A. Bogdanov, T. Isobe, *et al.*, "Midori: A Block Cipher for Low Energy", in *ASIACRYPT*, ser. LNCS, vol. 9453, Springer, 2015, pp. 411–436.

[198] C. Beierle, G. Leander, A. Moradi, and S. Rasoolzadeh, "CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks", *IACR Trans. Symmetric Cryptol.*, vol. 2019, no. 1, pp. 5–45, 2019.

[199] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak sponge function family main document", *Submission to NIST (Round 2)*, vol. 3, no. 30, pp. 320–337, 2009.

[200] T. De Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen, "Masking aes with shares in hardware", in *International Conference on Cryptographic Hardware and Embedded Systems*, Springer, 2016, pp. 194–212.

[201] H. Groß, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order", *Cryptology ePrint Archive*, 2016.

[202] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: A very compact and a threshold implementation of aes", in *Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30*, Springer, 2011, pp. 69–88.

[203] F. Yao, H. Chen, Y. Wei, E. Pasalic, F. Zhou, and L. Fan, "Optimizing aes threshold implementation under the glitch-extended probing model", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[204] C. Beierle, G. Leander, A. Moradi, and S. Rasoolzadeh, "Craft: Lightweight tweakable block cipher with efficient protection against dfa attacks", *IACR Transactions on Symmetric Cryptology*, vol. 2019, no. 1, 2019.

[205] A. R. Shahmirzadi and A. Moradi, "Re-consolidating first-order masking schemes: Nullifying fresh randomness", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 305–342, 2021.

[206] A. Moradi and T. Schneider, "Side-channel analysis protection and low-latency in action: –case study of prince and midori–", in *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22*, Springer, 2016, pp. 517–547.

[207] N. Müller and A. Moradi, "Robust but relaxed probing model", *Cryptology ePrint Archive*, 2024.

[208] E. Prouff, M. Rivain, and R. Bevan, "Statistical analysis of second order differential power analysis", *IEEE Transactions on computers*, vol. 58, no. 6, pp. 799–811, 2009.

[209] 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)*, 2009.

[210] S. M. Trimberger and J. J. Moore, "Fpga security: Motivations, features, and applications", *Proc. IEEE*, vol. 102, no. 8, pp. 1248–1265, Aug. 2014, ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2331672.

[211] A. Duncan, F. Rahman, A. Lukefahr, F. Farahmandi, and M. Tehranipoor, "Fpga bitstream security: A day in the life", in *2019 IEEE International Test Conference (ITC)*, IEEE, 2019, pp. 1–10.

[212] S. Mal-Sarkar, A. Krishna, A. Ghosh, and S. Bhunia, "Hardware trojan attacks in fpga devices: Threat analysis and effective counter measures", in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, 2014, pp. 287–292.

[213] 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*, Chicago, Illinois, USA: ACM, 2011, pp. 111–124, ISBN: 978-1-4503-0948-6. DOI: 10.1145/2046707.2046722. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046722.

[214] E. De Mulder, P. Buysschaert, S. Ors, *et al.*, "Electromagnetic analysis attack on an fpga implementation of an elliptic curve cryptosystem", in *EUROCON 2005-The International Conference on" Computer as a Tool"*, IEEE, vol. 2, 2005, pp. 1879–1882.

[215] S. Tajik, H. Lohrke, J.-P. Seifert, and C. Boit, "On the power of optical contactless probing: Attacking bitstream encryption of fpgas", in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1661–1674.

[216] E. Peterson, "Developing tamper-resistant designs with zynq ultrascale+ devices", *Xilinx Application Note*, 2018.

[217] M. Ender, A. Moradi, and C. Paar, "The unpatchable silicon: A full break of the bitstream encryption of xilinx 7-series {fpgas}", in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1803–1819.

[218] H. Li, G. Du, C. Shao, L. Dai, G. Xu, and J. Guo, "Heavy-ion microbeam fault injection into sram-based fpga implementations of cryptographic circuits", *IEEE Transactions on Nuclear Science*, vol. 62, no. 3, pp. 1341–1348, 2015.

[219] A. Lesea, S. Drimer, J. J. Fabula, C. Carmichael, and P. Alfke, "The rosetta experiment: Atmospheric soft error rate testing in differing technology fpgas", *IEEE Transactions on device and materials reliability*, vol. 5, no. 3, pp. 317–328, 2005.

[220] V. Pouget, A. Douin, G. Foucard, *et al.*, "Dynamic testing of an sram-based fpga by time-resolved laser fault injection", in *2008 14th IEEE International On-Line Testing Symposium*, IEEE, 2008, pp. 295–301.

[221] A. Proulx, J.-Y. Chouinard, A. Miled, and P. Fortier, "Analyzing the vulnerabilities of external sdram on system-on-chip field programmable gate array devices", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2024.