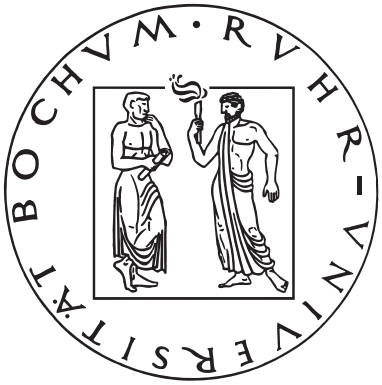
Advanced Methods for Hardware Reverse Engineering



Dissertation

---

zur Erlangung des Grades eines Doktor-Ingenieurs
der Fakultät für Elektrotechnik und Informationstechnik
an der Ruhr-Universität Bochum

---

by Christian Kison
Bochum, Mai 2018

To my beloved family.

Christian Kison
Place of birth: Soltau, Germany
Author's contact information:
Christian.Kison@rub.de
www.rub.de

Thesis Advisor:	**Prof. Dr.-Ing. Christof Paar**
	Ruhr-Universität Bochum, Germany
Secondary Referee:	**Prof. Dr. Jean-Pierre Seifert**
	Technische Universität Berlin, Germany
Thesis submitted:	30.05.2018, 2018
Thesis defense:	10.07.2018, 2018
Last revision:	July 5, 2019

Everything not saved will be lost.
–Nintendo "Quit Screen" message

Abstract

In daily life, it has become impossible to avoid electronic systems in our surroundings. We encounter them daily either in direct form, such as a Personal Computer (PC) and a laptop, or indirectly as part of an embedded system, such as mobile phones and key locking systems. In our modern world, we can use the electronics in our everyday activities to communicate our thoughts, ideas, and feelings over digital channels and subconsciously in routine work. They all have the common goal of creating a more comfortable life for us. Accordingly, these technological advancements and innovations leave a substantial user-specific digital footprint that holds daily routine and personal information that requires protection.

Consequently, innovations come with a risk if immature designed cryptographic systems are applied. We could leak confidential data, industrial expertise, or military insights. These risks become threats in the presence of malicious adversaries that try to exploit or even create system weaknesses for personal gains. Attackers can bypass embedded system security or create security vulnerabilities at the very root by altering any core Integrated Circuit (IC) component. For example, they could modify the circuitry during the design or fabrication. Malicious circuity, known as hardware Trojans, or vulnerability discovery can invalidate sophisticated security measure or deny the IC's service. They pose a major threat to today's military systems, financial infrastructures, transportation security, and safety of household appliances. Even though advanced countermeasures against hardware Trojan insertion by an untrusted offshore fab exist (e.g., split manufacturing, Trojan-aware hardware design techniques), they are not a panacea since they might be circumvented. Thus, detection of inserted hardware Trojans is the only reliable mechanism to witness malicious circuitry. Since many resources are required to analyze and detect hardware Trojans in manufactured ICs (by using visual inspection), it is believed that only few institutions can actually address such concerns. This limits sophisticated institutions to perform further threat assessements of advanced hardware trojans e.g physical parasitic Trojans.

The focus of this thesis is to enhance the Hardware Reverse Engineering process for the scientific community and tackle the existing prejudice that invasive hardware reverse engineering is expansive and is only possible within wealthy institutions. We ease the deprocessing that is necessary for hardware reverse engineering that enables us to carry out more hardware security audits. Furthermore, we want to show additional steps for to detect advanced parasitic hardware Trojans that are hard to detect with known academic techniques.

The first project, *Security Implications of Voltage Contrast*, introduces a new Region of Interest (ROI) detection method with the Scanning Electron Microscope (SEM) in a Voltage Contrast (VC) analysis. This technique allows us to gain valuable knowledge about the underlying hardware in a frontside and backside attack, which is beneficial for the hardware reverse engineering. Furthermore, we use the VC analysis in a new side channel attack for key leakage that designers and manufacturers must consider for future hardware protection. We use

an XMEGA32A4U as our Device Under Test (DUT). The second project covers *Security Implications of Intentional Crosstalk* and evaluates the security of a new parametric Trojan, the crosstalk Trojan. In a proof of concept, we design an intentional crosstalk on a publicly available AES core that triggers secret key leakage and in the fully-fledged OR1200 processor to leverage user privileges. We discover that current hardware reverse engineering cannot cover this kind of malicious circuitry and extend the known process to mitigate the former. In the third project, *Hardware Reverse Engineering of complex CPU-Microcode* we target a modern fully fledged AMD K8 CPU to reverse engineer the microcode and extract the microcode ROM. In this thesis the focus is set on the microcode extraction that allows to gain knowledge of the microcode engine. We reveal the complex, highly optimized ROM architecture of a sophisticated, optimized CPU and use the knowledge gained to construct our own microcode updates.

The results of this project shows that we are able to gain valuable knowledge of even complex CPU internals and proprietary designs with common reverse engineering equipment. Furthermore, hardware reverse engineering can be extended to ease the planar deprocessing to a significantly smaller ROI while gaining valuable knowledge for any reverse engineering. Thus, much effort can be eliminated when using further techniques e. g., from the Failure Analysis (FA) community as support. Furthermore, we address the problem of parametric hardware Trojan detection and demonstrate a mitigation of the crosstalk Trojan that was introduced.

Keywords

Hardware Reverse-Engineering, Side-Channel, Physical Attack, Invasive Attacks, Hardware Trojans.

Kurzfassung

Fortgeschrittene Methoden für Hardware Reverse Engineering

In unserem täglichen Leben ist es unmöglich, den Umgang mit elektronische Systeme in unserer Umgebung zu vermeiden. Wir begegnen ihnen täglich entweder in direkter Form als PC und Laptop oder indirekt als Teil eines eingebetteten Systeme, wie zum Beispiel Mobiltelefonen und digitale Schlösser. Wir nutzen sie in unserem Alltag um unsere Gedanken, Ideen und Gefühlen über digitale Kanäle auszutauschen. Sie alle haben das gemeinsame Ziel, uns ein komfortableres Leben zu ermöglichen. Dementsprechend hinterlassen all diese Technologien und Innovationen einen digitalen Fingerabdruck unseres Alltages und unserer Person, der abgesichert sein muss.

Folglich bringen diese technologischen Fortschritte und Innovationen leider oft auch Sicherheitsrisiken mit sich, wenn unausgereifte kryptografische System zum Einsatz kommen. Wir könnten geheime Informationen, industrielle Geheimnisse oder militärsiche Vorgänge unabsichtlich verraten. Diese Risikien werden zu Bedrohungen, wenn Angreifer mit böswilligen Absichten anwesend sind, die Sicherheitsmechanismen eingebetteter Systeme direkt an der Wurzel aushebeln, indem sie auf unterster Ebene Chip-Komponenten ändern und zum Beispiel Integrierte Schaltungen während des Entwurfs und der Fertigung modifizieren. Böswillige Schaltungen, die auch als "Hardware-Trojaner"bezeichnet werden, können hochentwickelte Sicherheitsmaßnahmen ungültig machen oder notwendige Dienste behindern. Sie stellen eine große Bedrohung für heutige militärische Systeme, die finanzielle Infrastruktur, die Transportsicherheit, sowie Haushaltsgeräte dar. Obwohl fortgeschrittene Gegenmaßnahmen gegen hardware Trojaner (Split-Manufacturing, Trojaner-bewusste Design Techniken) existieren, sind diese kein Allheimittel da diese umgangen werden könnten. Damit bleibt nur die Detektion von ggf. eingesetzten Hardware Trojanern, durch visuelle Inspektion als verlässlicher Erkennungsmechanismus. Da für die Analyse und Erkennung von Hardware-Trojanern in Endprodukten (durch visuelle Inspektion) viele Investitionen und Ressourcen benötigt werden, glaubt man, dass nur wenige Institutionen solche Probleme lösen können.Folglich können nur wenige Institutuionen eine Gefahrenabschätung von Hardware Trojanern durchführen, besonders für weiterführende hardware Trojaner, beispielsweise parasitäre hardware Trojaner.

Der Fokus dieser Arbeit liegt auf der Verbesserung des Hardware Reverse Engineering Prozesses bei der Analyse von Systemen mit komplexen kleinen Hardwarestrukturen für die wissenschalftliche Gemeinschaft. Wir erleichtern die Deprozessierung der Hardware, welche für Hardware Reverse Engineering generell benötigt wird. Außerdem zeigen wir zusätzliche erforderliche Schritte um fortgeschrittene parasitäre Trojaner erkennen zu können.

Das erste Projekt, *Sicherheitsimplikationen von Spannungskontrast*, zeigt eine neue Erkennungsmethode, mit der ein kleiner interessanter Bereich mithilfe des Rasterelektronenmikroskop in einer Spannungskontrast-Analyse identifiziert werden kann. Wir verwenden eine Spannungskontrast-Analyse in einem neuen Seitenkanalangriff, die den geheimen Schlüssel ermittelt. Gleich-

zeitig ermöglicht sie durch Vorder- und Rückseiten-Angriffen wertvolle Erkenntnisse über die zugrunde liegende Hardware zu gewinnen. Der XMEGA32A4U ist der hierbei eingesetzte Versuchschip. Das zweite Projekt behandelt *Sicherheitsimplikationen beabsichtigter kapazitiver Übersprecher* und untersucht einen neuen parametrischen Trojaner, den Crosstalk-Trojaner. In einem Proof of Concept konzipieren wir ein beabsichtigtes Übersprechen auf einer internen Leiterbahn in einem öffentlich verfügbaren AES-Modul, in welchem der Trojaner einen geheimen Schlüssel offenbart. In einem weiteren Versuch mit einem modernen vollwertigen OR1200-Prozessor erhöht er bei gezielter Auslösung die Rechte eines Nutzers auf Adminrechte. Wir zeigen, dass der derzeitige Hardware-Reverse-Engineering Vorgang diese Art bösartiger Schaltungen nicht abdeckt, und erläutern die nötigen Erweiterungen für den bisher bekannten Prozess. Im dritten Projekt, *Hardware Reverse Engineering eines komplexen CPU-Microcodes*, haben wir eine moderne, vollwertige AMD K8-CPU untersucht, um den darauf verwendeten Mikrocode zu ermitteln und den Mikrocode-Speicher zu extrahieren. Wir zeigen die komplexe, hochoptimierte ROM-Architektur einer vollwertigen kommerziellen CPU und nutzen das gewonnene Wissen, um eigene Microcode-Updates zu erzeugen.

Die Ergebnisse der Projekte zeigen, dass auch moderne, komplexe CPU-Interna und proprietäre Designs erfolgreich analysiert werden können. Der Hardware Reverse Engineering Prozess kann erweitert werden, um die planare Verarbeitung auf eine signifikant kleinere Regionen zu begrenzen, was den Vorgang erleichtert und gleichzeitig ermöglicht, wertvolles nützliches Wissen für den Reverse-Engineer zu gewinnen. Wir zeigen, dass viel Aufwand erspart werden kann, wenn weitere unterstützende Techniken aus der Failure Analysis verwendet werden. Darüber hinaus befassen wir uns mit dem Problem der Erkennung von parametrischen Hardware-Trojanern und demonstrieren eine Abschwächung des eingeführten Crosstalk-Trojaners indem wir den Stand der Technik des Hardware Reverse Engineerings erweitern.

Schlagworte

Hardware Reverse-Engineering, Seitenkanalangriffe, Physikalische Angriffe, Invasive Angriffe, Hardware Trojaner

Acknowledgements

This thesis is the result of the last four years, which I spent at the Chair for Embedded Security at the Ruhr-University Bochum and the Bundeskriminalamt in Wiesbaden. I've visited conferences, workshops and had hackathon-like meetings all over the world, meeting people with the same interests which were often also their hobbies. In the last four years, I traveled in more-or-less regular intervals between Wiesbaden and Bochum, balancing my work life, and social life with sports and family visits "in the North", as often as possible. Wherever I was headed I felt like having a second, third or even temporary fourth home. It was a fun time of meeting and working with people, having a slightly different mindset compared to students in the north, where i am from. Here, I would like to express my gratitude and thank those, who made all of this possible and enjoyable.

First and foremost, I would like to thank my family for all of the support throughout the years. I know that I can always count on them and ask for their support and help, being a safe-net and backbone. Thank you for all your support, faith, love and the courage to challenge myself in such a remote place. In Wiesbaden, I would like to thank my friend and colleague Jürgen Frinken, who fought the BKAs bureaucratic hurdles with me. Thank you for introducing me to this topic, the inspiring work and the experience throughout the years. Back in Bochum, I would like to thank my flatmates Benjamin, Phillip and Marc for the great time we had. Knowing Marc back from the days in Brunswick, we first joked on taking this path and now finished it. I have to admit that it felt great beating him constantly (even drunken) in Mario Kart.

Coming back to academia, I am very grateful to my supervisor, Christof Paar. Aside from the scientific guidance and the helpful advises, you always managed to motivate me. I am very grateful for the wonderful working atmosphere at our chair and want to thank my colleagues and friends. Special thanks go out to my escape room mates Basti and Max for challenging every room record with me and Christian Zenger for the "freibeuter" evenings.

Furthermore, I would like to thank Johann Machnik, Gerhard Wagner and my colleagues at KT52 for being helpful with their vast experience and knowledge whenever I needed an advice. Also thanks to the BKA badminton team, the MTV1817 volleyball team and the RUB Europahaus for a good counterbalance to my work life.

I would also like to thank further co-authors (in alphabetic order) for the joint research work: Gunnar Alendal, Omar A. Awad, Georg G. Becker, Malte Elson, Robert Gawlik, Thorsten Holz, Nikol Rummel and Sebastian Wallat. A very big "thank you" goes to those (un)lucky enough to proof-read my thesis in the various stages of writing: Jürgen Frinken, Erik Krupicka, Daniel Kison, Marius Eggert and Dennis Grubert. Last but not least, I want to thank our team

assistant, Irmgard Kühn, who manages so many of the administrative tasks, keeps it off our backs, and is in general great, friendly person to talk to.

Table of Contents

Part I

Preliminaries

Chapter 1

Introduction

In our daily life, it has become impossible to avoid electronic systems in our surroundings. We encounter them in the direct form, such as a PC and laptop or indirectly, as part of an embedded system, such as mobile phones and key locking systems with the common goal to create a more comfortable life. We use them in our everyday activities to communicate our thoughts, ideas, and feelings over digital channels or subconsciously in routine work.

With *Home automation*, we allow digital systems to control one of our most important and intimate zones. They can observe the house's inside temperature, open windows, and stove. They can even automate daily tasks, such as starting the heater, opening rolls or the door without being physically near the house.

Sensors and actors in the *car environment* are responsible for our entertainment and safety while some applications allow piloting on the driveway. They have become inevitable in today's fuel usage optimization and especially in critical safety applications, ranging from airbag control to smart break assistance in pedestrian protection. They can even make ethical decisions when choosing crash prevention courses.

Safety critical applications also appear in other fields, such as *health care*, *aerospace*, and *industry applications*, surveying potential dangers e. g., in hazardous environments. Even non-economic areas, like the military, space exploration, and political parties depend on electrical systems, such as digitally controlled military equipment and poll systems. They are increasingly accepted in a number of countries. These systems often enjoy such high levels of trust that they have become a pillar of our society.

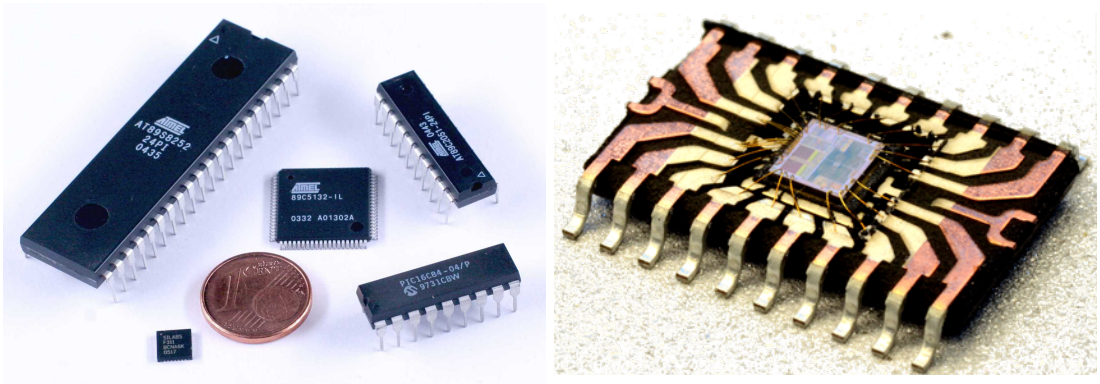


Figure 1.1: Multiple packaged micrcontroller ICs(left) and a depackaged IC(right).

The core of almost any embedded and digital system is an IC or a more advanced Central Processing Unit (CPU). Some exemplary microcontrollers ICs are shown in Figure 1.1. Depending on their applications, we have systems with varying complexity. While small power-saving Microcontroller (uC) can be used in small home automation projects, other more complex systems, such as modern cars, have more than 100 controllers. Computationally expensive ICs like CPUs and Graphics Processing Units (GPUs) are used in servers and desktop machines. Between them reside the embedded systems with a fluid transition on both sides. With today's' smartphone calculating powers, it has become hard to count them as a part of an embedded system or a fully hand-held desktop machine. Cars have also become increasingly powerful with complex automatic pilot/driving applications. Even Application Specific Integrated Circuits (ASICs) as dedicated hardware chips for one specific task, containing a static circuit, are included.

The increase in computational power has allowed artificial intelligence with neuronal networks to reemerge, benefiting from dedicated ASICs and complete human tasks in fascinating ways. Various scientific fields, such as biology, medicine, and computer science apply specialized neuronal networks in their research. Another miscellaneous application, *GO*, known as the last board game with a human in the lead, has been conquered with neuronal networks running on Tensor Processing Unit (TPU) ASICs. An ever-increasing demand for new IC designs and solutions exists.

IC Design High-volume IC production is based on a vast, globally distributed network of designers, vendors, and fabrications. Designers and vendors are economically driven to optimize their design and use the latest fab while the fab pushes to ever smaller technology sizes. Starting with *Moore's law* from the 1970s, it was estimated that Complementary Metal Oxide Semiconductor (CMOS) technology size shrinks by half every 18 months. Hereby, the width of the transistor gate is the naming factor of the technology node, even though in modern technologies, this is not always true. In fact, in the 14 nm technology size introduced by Intel in 2017, not a single element is close to the size of 14 nm. This might be an indicator that Moore's law might no longer hold true. Nevertheless, the industry continues the naming convention of the technology (half-)nodes.

The IC industry earned 412.2 billion USD in the year 2017. International Roadmap For Devices And Systems (IRDS)'s estimations for the first ICs with 7 nm technology are expected in 2019, while 5 nm is expected to be produced in the year 2021. Each shrinkage introduces new problems and further requirements on the purity of the materials. New equipment requires constant parameter fine-tuning while maintenance is enough to reset all efforts. The requirements for further technology nodes can be illustrated with the actual size under 10nm: this gate width is equivalent to less than 100 atom layers. Small impurities can easily result in one defective transistor out of the billions rendering the sample useless. Any defect, such as a transistor that does not meet speed or power constraints, results in a worse yield. Hence, new technology is only enrolled when it results in an economic advantage. To this extent, the vendors introduce ever more sophisticated techniques, like 3-dimensional gates (FinFET), Silicon-on-Insulator (SOI) processes, hyperscaling, and UV light lithography, driving the price for a new fab to a new height of 8.5 billion USD.

The yield of the fab is a direct factor for the IC industry's margin, causing it to become the most important factor for the economic rentability. This sets the resources for yield improve-

ments as basically unlimited. To underline its importance, the IRDS even has its own *yield enhancement* group, optimizing solely this field. The specialized process for finding, identifying, and preventing IC faults during the fabrication is the **Failure Analysis (FA)**. By creating a process of analyzing faults and fine-tuning the fabrication, the yield has steadily improved. FA research is presented yearly at the ISTFA[1] conference.

Failure Analysis (FA) techniques are dedicated to locating and isolating the faults produced during the fabrication or further preparations. They include all production steps from a large area fault during packaging to small cracks in the gate isolation or impurities in the wafer. Please note that FA techniques are applied from the vendor with all the necessary information of the layout, a debugged interface and simulations. They do not include basic design choices prior to simulations. If the fault cannot be explained by the extended debugging interface in a chip, first non-invasive techniques for a coarse ROI localization can be used, such as measuring side-channels, like heat, photonics, electro-magnetic (EM) emanation, and power consumption. For fine localization advancement, more invasive techniques e. g., Laser Voltage Probing (LVP), VC, and Atomic Force Microscopy (AFM), which exploit the physical characteristics of the CMOS technology, are considered. Once their results are compared to simulated data from Electronic Design Automation (EDA) tools, the ROI location can be determined with high accuracy. Sometimes, multiple techniques can be cascaded to further pinpoint the ROI.

Once the fault has been isolated, the last steps of the precise sample preparation are completed with a Focused Ion Beam (FIB) system to determine the exact fault. The FA has the major drawback that only a small number of samples—or even just one unique sample—exist. Deprocessing in a small technology size remains a challenge. Sample preparation requires expertise and necessary equipment to reliably reveal the faults. Please note that it might be necessary to prepare the sample for the FA technique(s) as well, without damaging the sample or excluding other techniques with this step.

FA processes have been well studied due to ample funding, and they are powerful tools in used to analyze the inner workings and state of the IC. They are useful to gain additional knowledge of any IC-internals. Since this highly specialized equipment is bought by the industry, it comes with a heavy price tag that academia is unable to spend. Furthermore, to fully utilize the FA techniques, in-depth knowledge from the devices is mandatory. If this knowledge is not present, the IC has to be reverse engineered.

Hardware Reverse Engineering: In general, the process of acquiring in-depth layout and schematic information of an unknown IC and its environment is called **Hardware Reverse Engineering**. It is the main focus of this thesis. In the context of an IC design supported FA, the engineer acquires necessary details from the designer. Nevertheless, in most cases, customers and third-parties do not have such a source of information, even though design internals of discontinued ICs are required.

Rapid advances and innovations result in short technology life cycles for hardware and software components. Once the hardware has a new revision, old ICs become obsolete and are discontinued. This becomes a problem when outdated hardware components wear out and further equipment depends on highly specialized, proprietary interfaces. They become unavailable

[1] International Symposium for Testing and Failure Analysis

for repairs or replacements since nanoscale chip repairs are generally very difficult. In the worst case, expensive industrial equipment, tools, and machines are on standstill due to a single defect IC component with unknown circuitry or some valuable data persists in inaccessible memory. Hence, the chip must be reverse engineered for reproduction or forensic analysis. Forensic applications can access lost (key-)data and determine evidence in law enforcement cases through hardware reverse engineering.

Special ASICs Intellectual Property (IP) cores are increasingly included during the IC design phase. They are inserted either as hard-wired or soft cores, depending on the buyers' license and preference. Hard-wired cores are delivered as already-routed layouts, while soft cores are placed and routed during the IC design and are usually more expensive due to their flexible options. Depending on the license, a certain number of copies or design changes are allowed. The hard-wired cores are usually performance-critically routed, or given unannotated to counter overproduction and intellectual theft. The already optimized design has been researched, developed, and tested. If enough knowledge has been generated, it is patented. In the presence of malicious competitors that try to cut their investments and illegally copy or overproduce an already finished design, the legitimate IP holder requires a means of control. Hence, hardware reverse engineering can be used to analyze the suspected designs and detect IP fraud. Interestingly, this shows hardware reverse engineering has two faces as it is first used to reverse engineer protected IP cores, but also for detection of the former.

In the cryptographic community, the risk of proprietary 'security-by-obscurity' systems is well known. Multiple examples from the recent past show that vendors sometimes design immature and dangerous cryptographic systems. They risk leaking personal data, industrial expertise, and military insights once the obscurity vanishes. Once the chip is deployed and a vulnerability is discovered, (hardware) patches are possible (in very few cases) and might force massive hardware rollbacks and damage the reputation. In order to protect against these risks, it is recommended to use well-known cryptographic primitives, and vendors need to certify security-related designs. Hardware security audits guarantee security by testing against non-invasive and semi-invasive attacks in certified laboratories. Reverse engineering can lead to the discovery of further weaknesses (e.g. Side-Channel Analysis (SCA) or invasive probing attacks) and improve security certificates.

Security risks become threats in the presence of malicious adversaries that try to exploit or create system weaknesses for personal gains. They can bypass embedded system security or create vulnerabilities at the very root by altering any core IC component, e.g. modify the circuitry during design or fabrication. In the worst case, an adversary starts tampering during the IC design phase and denies further security patches after roll-out. Malicious circuity *a.k.a.* known as 'hardware Trojans' can invalidate sophisticated security measures or deny the IC's service. Therefore, they pose a major threat to today's military systems, financial infrastructures, transportation security, and household appliances [143]. Since ample resources are required to analyze and detect hardware Trojans in manufactured ICs (by using visual inspection), only a handful of institutions can actually address such concerns [143].

With further advances in Trojan design, only invasive hardware reverse engineering strategies can counter such circuits, particularly when no golden model is present, or the golden model

has to be verified [145, 143]. Nevertheless, the search for countermeasures has become of increasing interest in academia. Embedded system security has become an important academic sub-discipline with high industrial participation. In particular, they attempt to (i) find respective malicious circuits [26], (ii) secure the IC design [153, 52], or (iii) secure the production chain [74, 15].

Ideally, the results of hardware reverse engineering provide a complete understanding of the underlying IC. With modern multi-billion transistor ICs, this is an infeasible task without automatic solutions. Even considering that tedious tasks can be automated, human interactions are required for understanding the resulting sea-of-gate. Thus, the task of hardware reverse engineering is a dedicated, specialized task, often taking several months of work. Reverse engineering only selected parts of the IC eases the process and saves resources. Once the exact implementation of the interesting parts, such as hardware Advanced Encryption Standard (AES) accelerator or memory encryption unit, is known, there are further non-invasive or semi-invasive attacks possible against which the designs are not protected.

Side Channel Analysis (SCA) Non-invasive attacks are attacks that do not change or alter the chip internally and externally. They observe the DUT during operation to attack the *implementation* of an algorithm through their observed side channels. These include heat, photonics, EM emanation, sound, and power consumption. These observations allow derivation of internal signals and secret key material with different leakage models and data hypotheses. This highlights that any algorithm running in the IC can be attacked with side channel attacks if the implementation is known and no generated on-chip randomness is used. Hence a-priory hardware reverse engineering is mandatory for unknown proprietary algorithms before they can be attacked with SCA. Semi-invasive die preparation improves existing side channels, such as the EM emanation. Otherwise, it creates access to side channels, such as the photon emission from the backside of the chip. Please note that side channels have also been used for hardware reverse engineering in Side Channel Analysis for Reverse Engineering (SCARE) attacks [106, 47, 60, 151, 10, 142, 68, 53, 40].

In the context of the three topics of the IC design, the failure analysis and side channel analysis interdisciplinary topics can support the academic topic of hardware reverse engineering. Figure 1.2 shows possible intersections with highlighted topics that are related to this thesis.

Aim and contributions of the thesis: The aim of this thesis is to improve hardware reverse engineering. The contributions of this thesis show that the current state of hardware reverse engineering (1) can be linked with other side-channel and failure analysis techniques. This can be done to help the hardware reverse engineer prior to the tedious gate-level netlist reverse engineering. It is (2) not sufficient to include advanced emerging parametric Trojans and (3) can be extended with a proposed process-flow to detect parametric Trojans. An overview with contributions (1) to (3) can be seen in Figure 2.2. Finally, this work (4) shows the reverse engineering process of an optimized mask NOR ROM in a modern, fully-fledged CPU to help reverse engineer its microcode engine and content. To show statement (1), we introduce the Voltage Contrast Side Channel Attack (VCSCA) that uses the VC to gain additional knowledge of the function of wires and gates within the sea-of-gates. In order to show the statement of (2)

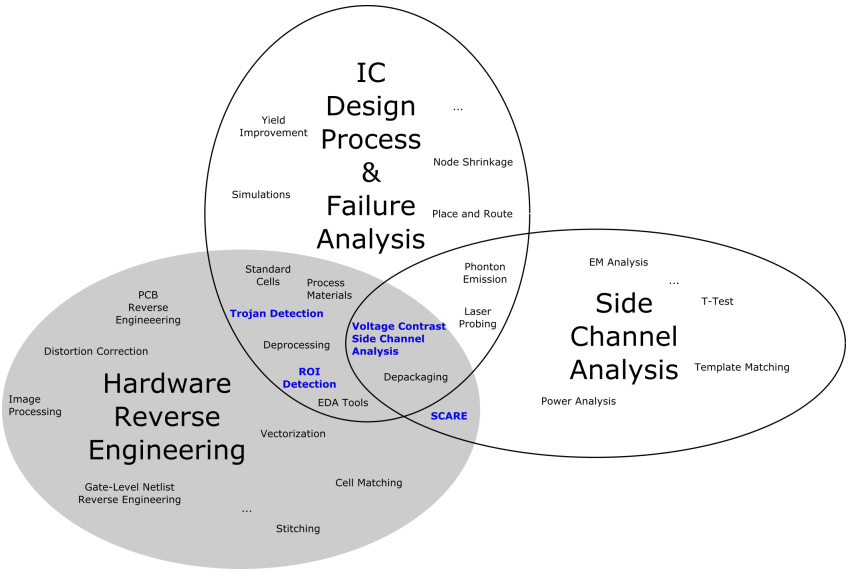


Figure 1.2: An incomplete intersection Diagram of Hardware Reverse Engineering, the Failure Analysis coupled with the IC Design and Side Channel Attacks point out some possible interdisciplinary topics.

and (3), this thesis presents a new parametric Trojan that is impossible to detect using state-of-the-art hardware reverse engineering and gate-level netlist reverse engineering. In fact, it is difficult to see the Trojan if the reverse engineer does not know of its existence.

Please note that this thesis focuses on the hardware aspect of deprocessing and die reverse engineering prior to the gate-level netlist. Some publications rename "hardware reverse engineering" as IC reverse engineering or *silicon reverse engineering* to highlight the sub-Printed Circuit Board (PCB) reverse engineering. Algorithms and techniques for post gate-level netlist reverse engineering have a sizeable academic community to automatically detect high-level abstractions, but are not the main part of this thesis. Another adjoining academic field camouflages gates to increase the reversing effort. Some publications have been done in academia, but during the time of this work, no real-world camouflaged gates emerged that would have an impact on the practical current, state-of-the-art hardware reverse engineering. Hence, both academic topics are mentioned and discussed in the respective chapters.

Structure: This thesis is divided into three parts, followed by an appendix. Part I consists of the preliminaries, including this introduction and the current state of hardware reverse engineering (Chapter 2). Part II presents the academic analysis of voltage contrast and capacitive crosstalk projects in detail. We start with a side channel attack against a real-world XMega32A4U in Chapter 3 to obtain a full key recovery of a dedicated AES core and review different methods of how the VC can support a hardware reverse engineer in his tasks (contribution(1)). Chapter 4 includes the implementation of a new kind of parametric hardware Trojan that exploits the capacitive crosstalk effect for malicious purposes (contribution (2), (3)). Chapter 5 in Part III covers the hardware reverse engineering of microcode mask ROM from a real-

world, fully-fledged and modern CPU (contribution (4)). The thesis concludes with suggestions for future work and closing remarks in Chapter 6.

Chapter 2

State-of-the-Art Hardware Reverse Engineering

In this chapter, we systematically survey hardware reverse engineering. To this end, we examine diverse methods and techniques to analyze Integrated Circuits (ICs) and Application Specific Integrated Circuits (ASICs) in order to retrieve the crucial (annotated) gate-level netlists of a hardware design in Section 2.1 and Section 2.2. The goal of the reverse engineer is to understand (parts of) the design's inner workings in order to perform another high-level task, i.e. to detect counterfeit products or to inject hardware Trojans. Therefore, we briefly discuss the state of the gate-level netlist reverse engineering (Section 2.3), which focuses on the retrieval of high-level Register Transfer Level (RTL) information (e.g. control unit or datapath components). Please note that the gate-level netlist can be obtained through several means in multiple real-world scenarios, i.e.

(1) chip-level reverse engineering (see Section 2.2),

(2) bitstream reverse engineering in case of Field Programmable Gate Arrays (FPGAs),

(3) directly from the layout in case of an untrusted (off-shore) foundry or from an Intellectual Property (IP) provider.

Note that this model is consistent with prior research on hardware security [9, 35, 116, 26].

A survey of anti-reverse engineering techniques is out of the scope of this work, but the interested reader is referred to [59].

2.1 PCB Reverse Engineering

The deprocessing of hardware reverse engineering starts at the Printed Circuit Board (PCB) and package level of the piece of hardware. Firstly, the IC is removed from the PCB e. g., through desoldering or drilling. The challenge to protect the die is becoming ever more difficult with reduced die size, Ball Grid Array (BGA) grid pitch, die thickness, and even underfilled flip-chip packages. Secondly, the mold package must be removed with wet-chemical or mechanical means. Hereby, the die is to be protected, which often results in choosing wet-chemical depackaging, as the die is protected by the seal-layer[1] from the front side. The backside offers enough silicon in bulk to withstand carefully applied depackaging processes as well. The bonding wires are of special concern, as newer copper bondings are, compared to gold bonding wires, difficult

[1] passivation, often SiO_2

to preserve. For invasive hardware reverse engineering, the wires can be neglected once their connectivity is known or the connectivity can be derived. Advanced techniques for finding bonding wire connectivity can be completed with (3D) micro-X-ray-tomography or selective packaging delayering with a mill.

2.2 Chip-level Reverse Engineering

To learn the gate-level netlist of an ASIC post-manufacturing, chip-level reverse engineering has to be performed when the gate-level netlist cannot be obtained through other sources. Here, the goal is to deprocess and image the IC layers until the IC is fully digitized and interpreted as an error-free gate-level netlist. To this end, various steps are involved: (1) depackaging and mechanical preprocessing, (2) delayering and imaging, and (3) software-based post-processing [108]. An overview can be seen in Figure 2.2. The hardware reverse engineering is coupled with the IC design process it is supposed to reverse. Common practices (e.g. routing styles, standard cell rows, ...) and optimizations(e.g. flipped standard cell rows, ...) have great potential to ease, correct, and optimize steps in the hardware reverse engineering process. They are indicated with dashed double arrows in Figure 2.2. Each step for hardware reverse engineering will be explained in more detail in the following. The interested reader is referred to [155] for details in the IC design process. A schematic of a stacked IC can be seen in Figure 2.1.

2.2.1 Depackaging and Physical Preprocessing

The first hurdle of chip-level reverse engineering is to depackage or decapsulate the molded chip to gain access to the physical design. This process depends on the package and is done through wet-chemical or mechanical means. In particular, the die must be protected from any harm. Typically, wet-chemical depackaging is chosen since the die is protected by a seal-layer from the front side (often a SiO_2 passivation layer). Note that the backside usually has a silicon bulk stabilizing the die to withstand careful depackaging processes. Additionally, bonding wires are of special concern during any (semi-)invasive attacks while the chip remains operational.

Special packages, e.g. fully embedded ceramic packages or metal cases, might be hard to etch. They have to be mechanically drilled with a powerful Computerized Numerical Control (CNC) drill or by hand. Please note that Cu bondings are of special concern for future work with the IC industry mitigating to Cu processes, for production cost reasons. After fully depackaging the die, preparation to retrieve the physical design from Figure 2.2 is started. Prior knowledge through Failure Analysis (FA) andVoltage Contrast Side Channel Attack (VCSCA) as introduced in Chapter 3 are very valuable at this point.

Deprocessing. Hardware reverse engineering through visual inspection is an invasive, destructive process used to extract functionality information from the ICs. Therefore, IC dice are thinned in an alternating delayering and scanning process to obtain an image of every stacked metal and polysilicon layer.

The delayering process is a combination of wet chemical, mechanical, and plasma processes (often summarized as Chemical Mechanical Polishing (CMP)) to optimize the planarity of the die and the contrast for the imaging step via Scanning Electron Microscopes (SEMs)[108, 145]. In particular, planarization of the current layer with a large *surface-to-thickness* ratio is challenging.

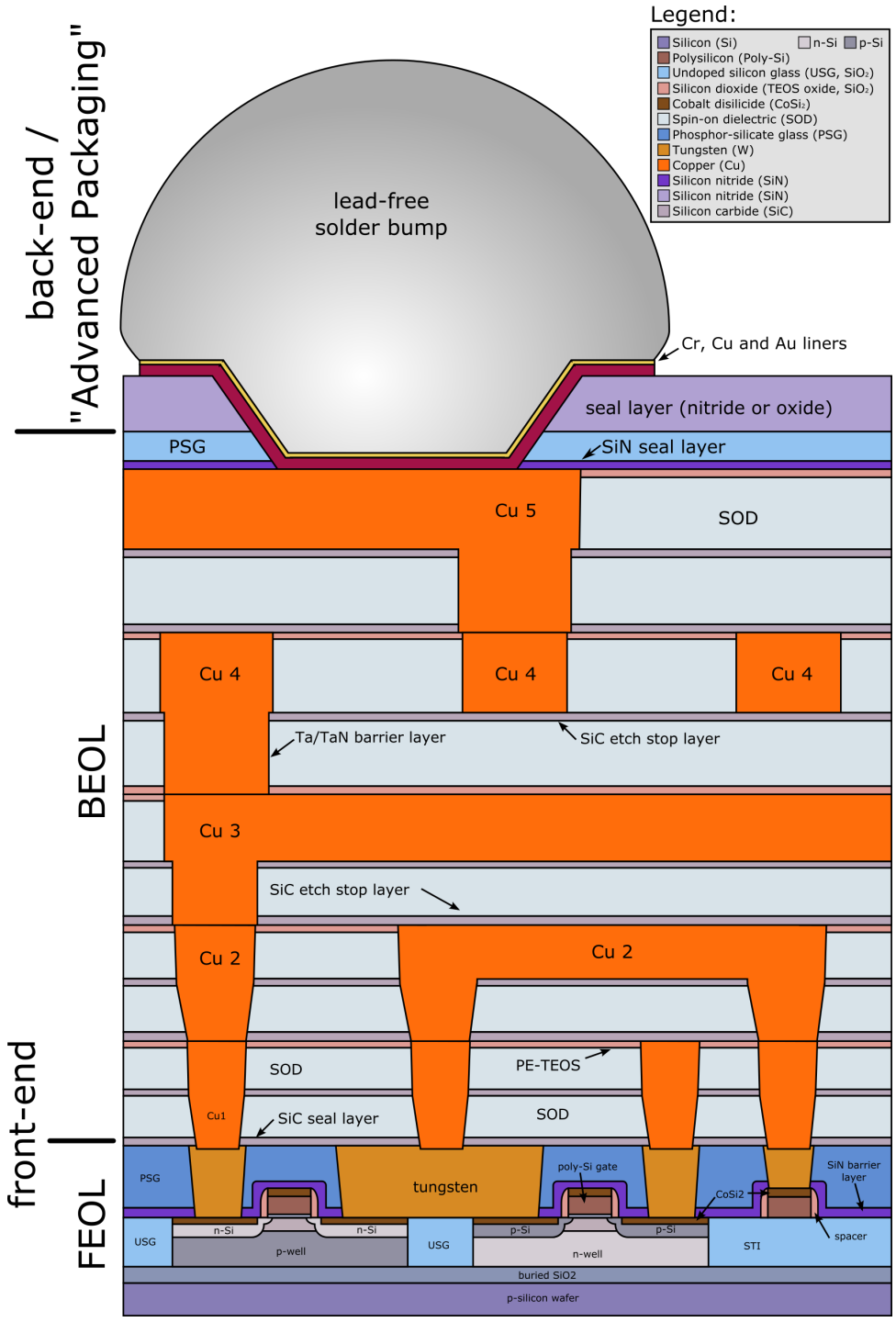


Figure 2.1: An overview of a stacked IC in a copper process. The front-end is built first with all the technology cells while the back-end interconnects respective cells. The schematic shown is a copper process with 5 metal layers and a SOI technology(see the buried SIO_2). Source in the list of Figures.

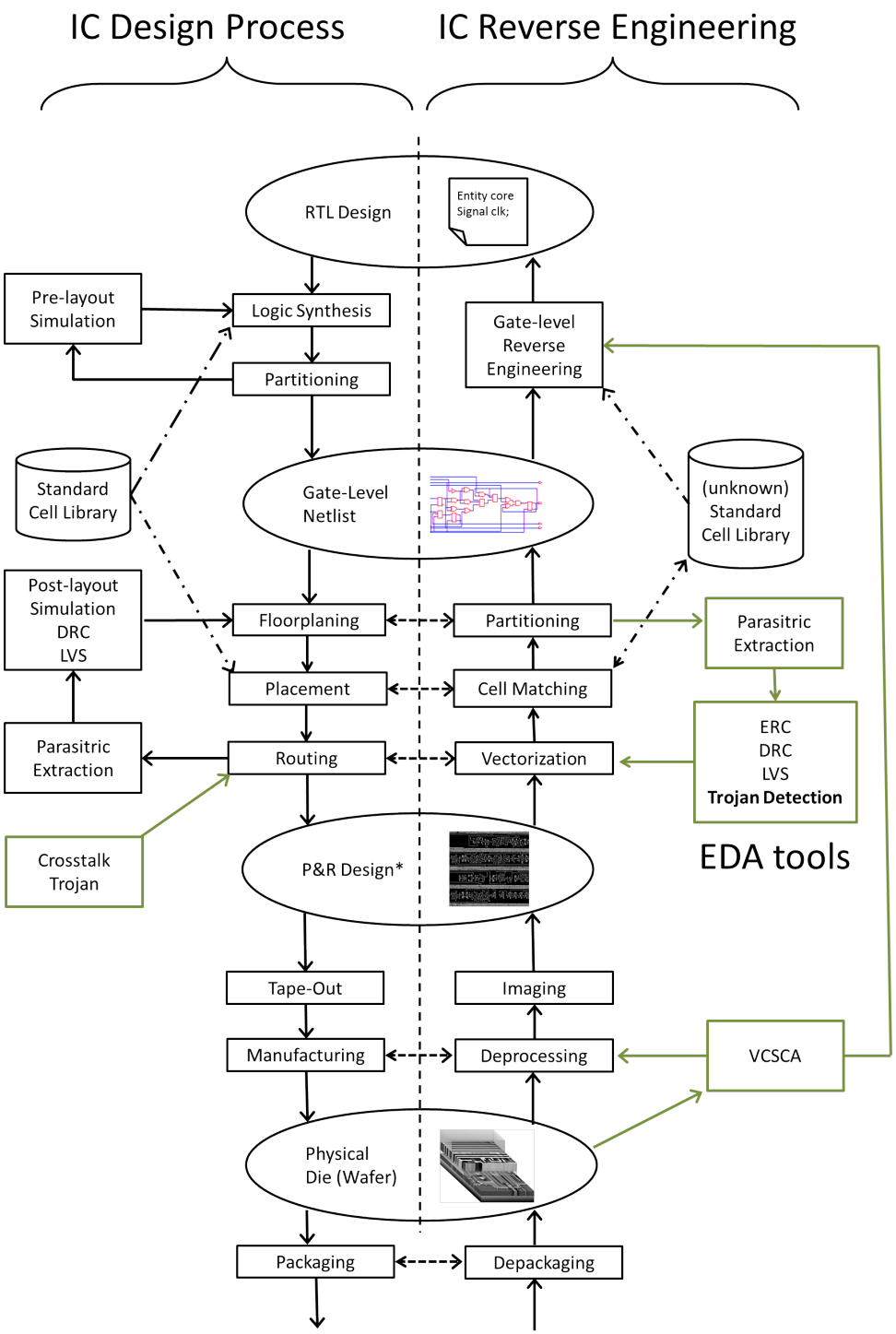


Figure 2.2: IC Reverse Engineering compared to the IC design process. LVS: Layout versus Schematic; DRC: Design Rule Check. *The Placed and Routed Design (P&R Design) includes cell instances for proprietary standard cells (for the fab) during the IC Design process, in contrast to unannotated designs during Hardware Reverse Engineering. Contributions of this thesis are shown in green.

Furthermore, it is very beneficial if the locations of the Region of Interest (ROI) are known, as the necessary planar surface is reduced significantly. The reverse engineer can then focus on his ROI while neglecting the rest of the chip [85]. These steps usually require expensive equipment and experienced users, as the shrinking technology node makes the process of planar delayering exponentially difficult. The risk to destroy the ICs under investigation always remains and requires multiple samples. The equipment resources increase with shrinking technology size, along with the necessary imaging time, precision, alignment, and calibration needs.

The complexity of deprocessing is further increased with current material changes to decrease costs and improve physical properties e. g., low K-materials and copper wire/via technologies. Each IC requires slight adjustments in equipment parameters and processing duration. In the following, we briefly introduce possible equipment.

Mechanical Polishing/Lapping tools are mandatory to polish in a "planar" way. It is mandatory to fix the sample on a flat surface prior to the polishing process. Two kinds of automated polishing methods are known: (1) is the lapping with a flat lapping stone while (2) is a very precise CNC drilling machine. Both types have adjustable spring forces used to press the sample with defined forces onto the lapping substrate. Surface alignment is done with a collimator or an accurate height measurement with a CNC-like setup. Please note that the IC surface can initially be curved due to packaging materials creating mechanical stress. Furthermore, as the lapping in the corners and edges is more intense (as a constant force is applied to a smaller area), a "bullseye" effect easily happens. This forces a reverse engineer to reduce the lapping time as much as possible and remove more material with other techniques.

Wet chemical processes have the major drawback of being isotropic, which can become harmful as deprocessing requires a planar surface in contrast to isotropic etching behavior. An example with two interconnects can be seen in Figure 2.3. Furthermore, the chemicals for etching (mostly fluoric-acid) SiO_2 are hazardous and toxic. It is not recommended to use them at all without the necessary safety gear. There are wet chemical processes used to selectively etch materials, such as Al and Cu. These can be used in one of the last steps, e. g., to etch Al to highlight the underlying Ti/TiN compared to the W vias shown in the imaging.

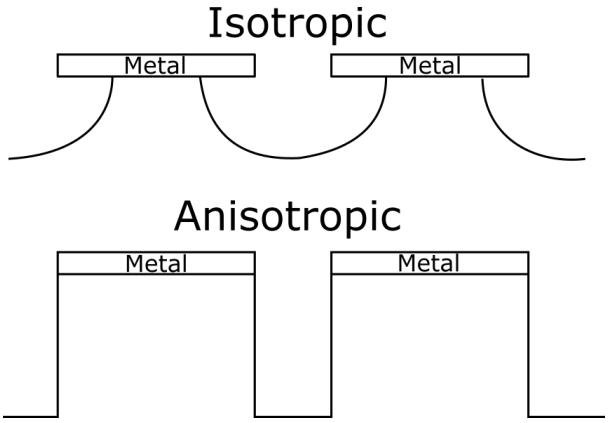


Figure 2.3: Isotropic compared to anisotropic etching. The isotropic etch might underetch structures, decreasing the stability. Anisotropic etching can done during plasma (dry) etching.

Dry/Plasma Etching is the process of etching with a plasma source, known as etch species. It can be either charged (ions) or neutral (atoms and radicals). A significant advantage to wet etching is the "directional", anisotropic etching which is more beneficial for planar processes. Plasma etchers are usually cut-off chambered machines where the $Fluorine^{+}$ radicals are fewer in numbers and exist only in a small chamber without human access during the processing. It is well researched in the context of the IC design process. Gases can be used to etch different materials selectively[21].

Broadband Ion Beams use the same physical principles as a Focused Ion Beam (FIB). Ions are accelerated to hit the surface of the IC at a very small angle but are not focused on a single spot or rastered. This physically knocks out ions from the surface of the sample in a slow process. As in the plasma process, different gas combinations can be used to specify a selectivity in the materials to be removed or speed up the process, imitating the plasma process.

Every IC has a different chip manufacturing process due to cost optimizations or technology node requirements. Therefore, different conductors, semiconductors, and dielectrics must be investigated and selectively removed without destroying the functional information of the IC [108]. Recipes from the IC fabrication or the FA are used or at least used as a starting point. For modern, nanoscale technologies, it is essential to use the necessary equipment to approximate or measure remaining layer thickness. Deprocessing quality can be assessed with invasive cross-sectioning or non-invasive optical-based thickness measurements.

Imaging In state-of-the-art reverse engineering, digitalizing and imaging is performed using a SEM or even a more eroding FIB. Since modern technology sizes hit the diffraction limit of optical microscopes, more advanced visualizing tools are mandatory. Such modern equipment is costly, but it results in higher magnification of images. A clear brightness yielded from the SEM images is beneficial for post-processing as it allows us to distinguish between vias, wires, and spin-on dielectric (SOD), see Figure 2.4. Usually, the sample has to be prepared to prevent the accumulation of electrical charges during the imaging process. The deposition of a very thin C surface layer through evaporation or low vacuum chambers are reasonable solutions. Further parameters, like acceleration voltage and precise stage movement, are beneficial for the following stitching and imaging processes.

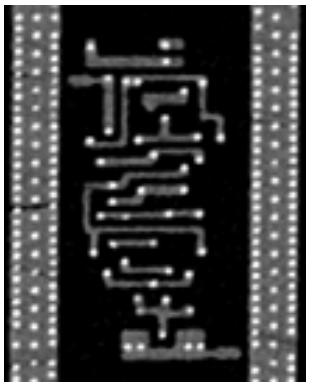


Figure 2.4: Example of metal 1 layer is shown. Brightness allows to distinguish between wires(light grey), vias, and the SOD(dark areas). The brighter dots are vias between Metal 1 and Metal 2.

An in-depth understanding of the physics of the processes and the necessary equipment are mandatory for achieving an adequate delayering quality. Basic recipes and processes are taken from the FA community as they have similar goals when isolating faults.

2.2.2 Software-based Post-processing.

In order to generate a functional chip representation from the digitized tile images (mosaic of a big image) of the previous step, the tile images have to be stitched and vectorized. This step can be separated into a Back End of Line (BEOL) and a Front End of Line (FEOL) process.

Vectorization/BEOL. To vectorize the BEOL, different image processing techniques have to be used to differentiate lines that are wires, circles and, dots(vias), as shown in Figure 2.4. In older process technologies, they appear due to the material mass differences of the aluminum wires and the tungsten vias. This difference can be used to separate the vias from the wires in a first step. Further geometrical constraints from the routing and placement geometries can be applied to improve the results.

Known wire extraction is based on edge detection[90] or grid based[78]. To improve the detection, image filters can use common geometrical design rules for vectorization. As the routing is commonly done in full λ steps, a constant minimal separation distance can be used to create a Design Rule Check (DRC) grid. Matching these with the underlying image allows for the detection and extraction of line segments. During the common routing process, two following layers have primary routing directions with a 90-degree shift. If the Metal-2 layer is routed horizontally, the next Metal-3 layer is routed vertically to create an orthogonal mesh, which is helpful for automatic line detection.

Cell Matching/FEOL. As most IC are built from a standard-cell library, containing around 50-300 cells, we first identify standard cells, including their rotated and mirrored versions to find their instances positions. Each standard cell is first recognized manually, and its function is reverse engineered once. Layers containing the active area, poly-silicon, and Metal-1 are necessary for manual cell reverse engineering. When the cell is identified, it can be automatically detected by using the images from the FEOL(e.g. poly-Silicon and M1) through image processing. As the (stitched) images are never free of faults, the image processing has to be robust and tolerant in order to recognize cell instances. Hence, pattern recognition, like template matching, is preferable. With robust cell image filters, the image can still be recognized with small defects(scratches, dust) and small routing differences (cell I/O, Metal-1 routing) in the image IC structures. Figure 2.5 shows the three standard cells that were detected.

Additionally, it is possible to automate the manual part in the FEOL recognition with Layout Versus Schematic (LVS)-like algorithms[51]. The images are first vectorized as in the BEOL process and in the following cell-parted in order to isolate single cell candidates. Then, a LVS algorithm is compared with each possible standard cell to find the function of the respective cell candidate. The authors of [51] automatically isolated and identified standard cell functionality with a database of known cell libraries. This LVS-like algorithm compares the transistors and its connections with annotated known transistor connection networks in a database. This allows us to identify cells, even if the cell library is from a different technology size or has different buffer strengths. Finally, with the standard cell instances in the FEOL and their connections through the metal lines in the BEOL, the ROI is extracted as a gate-level netlist.

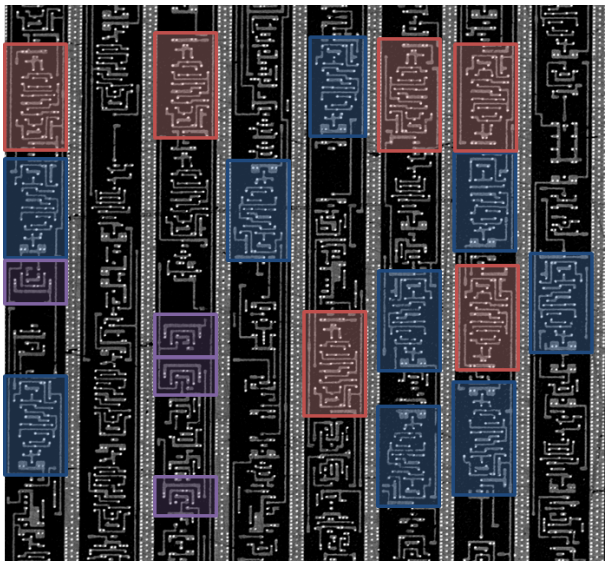


Figure 2.5: Cell Recognition of three standard cells on Metal-1. Small scratches in the bottom part and routing optimization on Metal-1 might disturb the recognition.Not that recognized cells are also mirrored and rotated.

2.3 Gate-level Netlist Reverse Engineering

After having specified how a reverse engineer can access the gate-level netlist for ASICs designs, we now provide an overview of publicly documented reverse engineering techniques to retrieve high-level information (e.g. control units or hierarchy information of submodules). Please note that gate-level netlist reversing techniques for FPGAs designs can be applied at this point as well.

Chisholm *et al.* [38] presented a workflow on how to reverse engineer module-level descriptions from gate-level netlists, addressing the synergy of the human analyst's creativity and the computer's ability to solve repetitive tasks. In a case study, Hansen *et al.* [71] described several best practices for a human analyst to reverse engineer gate-level netlist. Shi *et al.* [127] evaluated a technique to automatically reverse engineer a Finite State Machine (FSM) from gate-level netlists. Meade *et al.* [100] extended this technique in order to retrieve the state transition function for the reverse engineered FSMs. In further work, Meade *et al.* [99] developed a technique to separate control unit registers from datapath registers. In order to automatically reverse engineer functional submodules in a larger hardware design, diverse techniques have been developed based on Boolean function analysis [126], pattern mining of simulation traces and model checking [92], module boundary identification [135, 134], and word-level structure identification [93].

Since the functional identification of subcircuits requires us to find the correct matching between known subcircuits and the subcircuit under inspection, a reverse engineer has to find the correct input permutation. To avoid this computationally demanding task, Gascón *et al.* [64] addressed this problem with a template-based solution.

Part II

Security Implications of Voltage Contrast and Intentional Interconnect Crosstalk

Chapter 3

Voltage Contrast Side Channel Analysis

This chapter demonstrates how the Scanning Electron Microscope (SEM) becomes a powerful tool for Side Channel Analysis (SCA) and Hardware Reverse Engineering through Voltage Contrast analysis. This enables an attacker to locate AES bit-wires in the top metal layer and thus, to recover valuable netlist information. An attacker gets a valuable entry-point to look for weaknesses or Intellectual Property (IP) in the AES circuit. The results were published in [85] and extended by a backside approach in Section 3.6. The publication is a joint work with Jürgen Frinken.

Contents of this Chapter

Contribution: *In the context of this project, my contribution is the initial idea, implementation and evaluation of the given approach.*

3.1 Introduction

A crucial part in hardware reverse engineering is to know the location of their ROI prior to their delayering process. Without this knowledge, the literal search for the needle in the haystack can become a major obstacle for the reverse engineer. ROI s are usually security-sensitive elements, e.g., fuse bytes, cryptographic algorithms or proprietary parts of an IC [81, 130]. We may have luck if vendors "mark" sensitive areas with a shield or we already know the basic structure from a similar IC within the same vendor family. However, this is an exception especially in today's multi-million gate IC s. Identifying crucial elements is often extremely difficult.

The classical approach is a stepwise delayering down to the silicon substrate while taking images from each layer. After assembling and overlaying the different layers, a hardware reverse

engineer can begin to interpret the logical cells to find his ROI. Even with semi-automated tools that help to recognize standard cells of a library and wirings, we need to attend and review the process of millions of gates to get a flawless netlist from the chip structure: There is no feedback mechanism that alerts of the reverse engineer about mistakes and even the slightest mistake in the image acquisition might lead to faulty connections and a completely different circuit behavior. To make things worse, it is difficult to completely planarize the die with low to medium prized equipment, resulting in bad layer images and therefore worse recognition rates with the current chip-area-to-layer-thickness ratio.

Pinpointing the delayering process to a ROI reduces the costs and processing time when the user can focus on keeping the structure of the ROI still recognizable, neglecting the rest of the chip. Another reason is to achieve better signal-to-noise-ratio with located EM-traces or make other (fault-)attacks possible [120, 121]. Sometimes this might even enable more advanced analysis like inter-gate side channel leakages as discussed in [137].

3.2 Related Work

In order to find the hot spot or ROI, multiple more practical approaches emerged, often in combination with use of a Side Channel (SC). The EM near-field cartography, thermal analysis and optical photon emission are the three most noteworthy examples [120, 121, 113, 19, 32, 58, 129]. All techniques are semi-invasive attacks[131]. They depackage the IC to strengthen existing side channels like EM emanation (to make pinpointed small loop measurements possible) and to gain access to additional SC s like photon emission and heat distribution. The mentioned semi-invasive techniques are possible from the top- and backside of the chip, whereby the backside approach usually needs additional equipment for milling and thinning the silicon bulk [58].

To be precise enough for locating the smallest activity areas, spatial located EM emanation cartography takes several hours. It finds multiple high Signal-to-Noise Ratio (SNR) spots along the power distribution, as the radiation is directly related to the power consumption. Furthermore EM emanation often results in better SNR from the frontside, as the probe can be moved closer to the emanation source [120, 121, 113]. Nevertheless this approach allows to find a ROI within μm range. Thus, we can can skip most parts of the chip and concentrate on the ROI. The drawback for EM near-field cartography is the long measurement time to scan the whole die surface and may show multiple hot spots.

In the case of the photon emission, the price of the equipment maps directly onto the measuring time and quality: An IR-range sensitive camera with good Quantum Efficiency (QE) and low dark-current costs several 10k USD. Transistors have a probability to emit a photon during a state transition which passes through the thinned silicon backside. A first mandatory preparation is the thinning of the silicon bulk on the backside. The basic idea is to capture the emitted photons and visualize them in a highly spatial resolute setup. This becomes a major obstacle for modern CMOS processes, due to the diffraction limit of IR light. Additional preparation steps after the thinning, like a Numerical Aperture Increasing Lens (NAIL) , might become necessary [144]. This extends the preparation time and bears the risk of breaking the chip. After a camera-dependent image integration time, we can find the activity of individual transistors during the loop execution of one code fragment [129]. By increasing the supply voltage during the interesting clock cycles, the authors enhance the IR photon emission probability to highlight

corresponding areas. Using this technique the authors of [124] successfully extracted an AES key by observing the memory access pattern of the `subbytes` routine. Meanwhile [58] pinpoints a Picosecond Imaging Circuit Analysis (PICA) attack on single transistors to find `xor` values. Finding the respective hot spots and transistors is done with an optical long time image integration like in [124, 129]. Once a ROI has been determined, PICA can be used to see transistor switches in high-frequency ICs.

Surface Liquid Crystal (LC) is a wide spread analysis method based on the thermal radiation for failure analysis in the industry. It is well established and can be exploited for ROI localisation from the attacker's point of view. LC s allow spatial resolutions of 4 μm and better[19]. For modern CMOS processes this approach has equal spatial limitations, due to the IR light diffraction limit. Therefore new approaches like the Fluorescent Microthermal Imaging (FMI) emerged which try to fluoresce light with shorter wavelength [19]. Furthermore did the authors of [46] try to extend the thermal hot spot detection to the IC backside. Depending on the camera, are thermal and photon emission failure analysis methods qualified to find the ROIs. Both approaches are usually done over multiple program executions to heat-up or gather enough emitted photons in a normal working chip. These failure analysis methods face big challenges and difficulties with the upcoming CMOS sizes and decreasing power supply voltages. The modern CMOS size is below the diffraction limit of IR light and the decreasing power supply voltage drops the probability of photons being emitted and reduce the produced heat due to smaller currents.

This chapter uses the SEM as advanced inspection tool. Access to a suitable SEM should not be problematic as bigger institutes and universities with mid-class laboratories usually own one, due to their distribution in many academic fields. Companies and private persons can find second hand SEM s for under 10k EUR or rent it on a hourly basis. A well-funded hardware reverse engineer usually has access to a SEM for taking layer images. Once the setup is built and the Device Under Test (DUT) is vulnerable to our approach, we are able identify the ROI faster as the EM cartography and have less sample preparation compared to the photon emission analysis. Furthermore we don't require to run the chip multiple times and have better spatial resolution than thermal analysis methods.
Our contribution in this chapter is the following:

(1) We demonstrate an approach to pinpoint the AES location within the XMEGA microprocessor with Voltage Contrast (VC) analysis.

(2) We exploit the VC as a SC and perform a full key-recovery in a template-attack and a Simple Side Channel Analysis (SSCA).

(3) Using the VC images reveals additional information of the AES circuit useful for a reverse engineer. Furthermore are we showing the potential of VC SC in a Side Channel Analysis for Reverse Engineering (SCARE) approach to find additional circuit netlist information.

(4) The results show a possibility to counter hardware-obfuscation and hardware protection and to verify parts of an extracted netlist.

The rest of this section is structured as follows: Section 3.3.1 describes the different VC analysis methods and their physical understanding. Section 3.4 locates the AES circuit with the common Dynamic Voltage Contrast (DVC) analysis. Section 3.5 is the main contribution that gives a Proof of Concept (POC) for the VCSCA. The DUT is a widespread Atmel XMEGA microcontroller. We show a template-attack and a SSCA approach to recover the full AES key. Section 3.8 concludes this work.

3.3 Background

In this section, we will introduce the required background information and reference to more detailed descriptions. We start with the voltage contrast in general and continue with the dynamic voltage contrast and Capacitive Coupled Voltage Contrast (CCVC), respectively.

3.3.1 Voltage Contrast

The SEM has become a powerful diagnostic tool during the last 60 years, used in many applications for IC inspection and failure analysis. When an electron beam-gun fires (primary) electrons on a scanning surface, secondary electrons are hit out of a solid specimen. These emitted secondary electrons have usually low energy (0 - 50 eV), which makes them easily detectable by using a positive electrical-field metal-plate as a detector. Out of the SEM images are the secondary electron images the most widely used, due to their ease of production and similarity to light microscope with improved depth of field [24].

During VC failure analysis, the natural negative charge of the electrons is used to view different voltage potentials, with the help of their electrical field and their direct influence on secondary electrons. Note that VC also works with positive ions in a FIB since only the difference in charge is important. Using VC with positive ions from a FIB achieves better results[1] and the voltage interpretation of brightness and darkness is reversed compared to the SEM VC [23, 24, 28]. VC analysis needs the chip to be depackaged to gain access to the die surface.

VC analysis can be classified into two categories, which are on the one hand the static VC methods, including Passive Voltage Contrast (PVC) and Active Voltage Contrast (AVC) and on the other hand the Dynamic Voltage Contrast (DVC).

3.3.2 Static Voltage Contrast

The static VC is performed on chips with a removed passivation layer or even partly delayered chips. Static VC is split in the two sub-techniques PVC and AVC. The PVC does not connect the DUT to any signals or voltages and thereby shows the charging up of floating gates and capacitances. It is often used to find shorts and imperfectly connected wires and structures during chip manufacturing. The DUT does not have to be functional anymore, allowing to take VC images of intermediate layers. Note that is possible to create floating structures by removing metal layers or by wire cutting with a FIB [115]. To pinpoint shorts and badly connected wires, the structures are split and analyzed separately by applying a voltage in the AVC.

[1] With individually optimized parameters

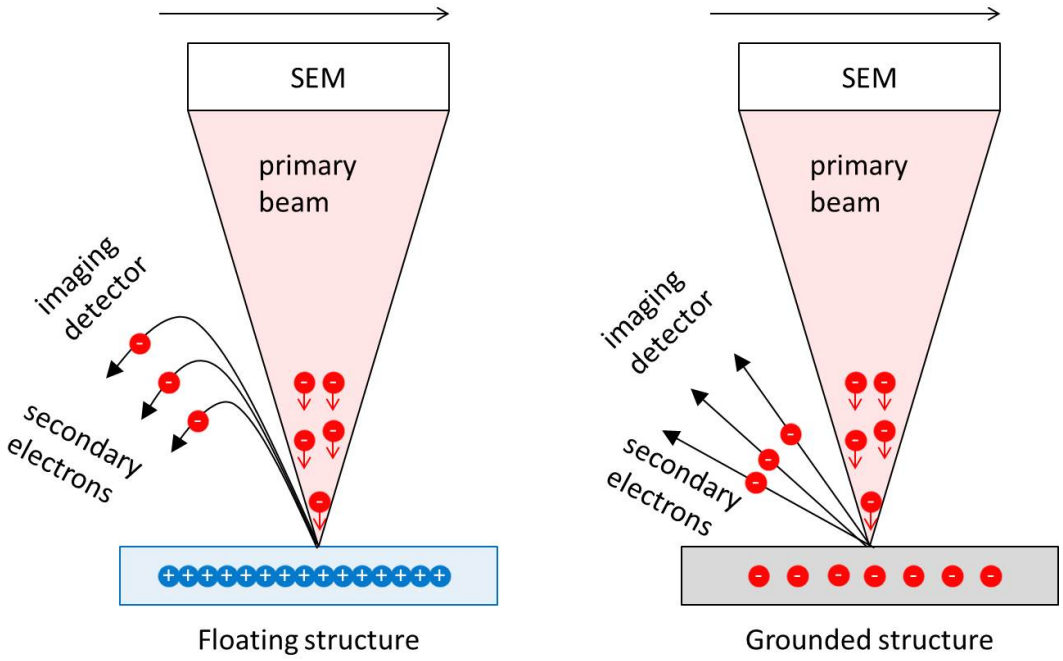


Figure 3.1: Passive Voltage Contrast. Floatin structures charge up while grounded structures are supplied with electrons from the GND signal.

Figure 3.1 shows the PVC. Isolated structures are charging up, due to secondary electrons being hit out of the structure. In the immediate consequence, the majority of produced secondary electrons are prevented to reach the detector by the inverted electrical field. These structures appear dark in the image. Grounded structures do not appear bright, because of the high yield[115].

AVC differs from PVC as it applies voltages in some structures to force them to look dark or bright in flawless structures. Is the outcome not as expected, a short or open connection can be assumed. Knowing the detailed place-and-route of the netlist and structures is very helpful, but not mandatory. The authors of [136] use the PVC to detect stealthy dopant-level circuits (trojans).

3.3.3 Dynamic Voltage Contrast

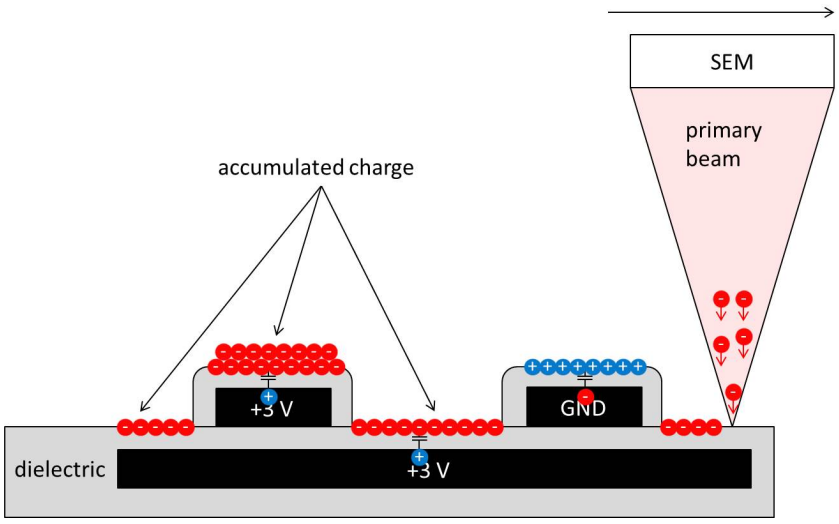
DVC is performed during dynamic rather than the static operation of the DUT. In the scope of an IC or Microcontroller (uC), the device is running normally while performing the voltage contrast. If the device is still under the passivation layer, CCVC can be applied. CCVC exploits the property that the voltage potentials of the top metal-wires are electrically coupled with the covering dielectric passivation of the die, forming a capacitor. Therefore, CCVC is performed while the passivation layer is still covering the die and the chip is still operational, or at least voltages can be applied to top wires and structures.

When a line or wire buried under the passivation is assumed to have the voltage U_p, an voltage U_S is generated through capacitive effects. This effect can be described as a transfer function U_S/U_p, which depends on the electrical (U_E) and geometrical (d_e, d_p, W) parameters. U_E and

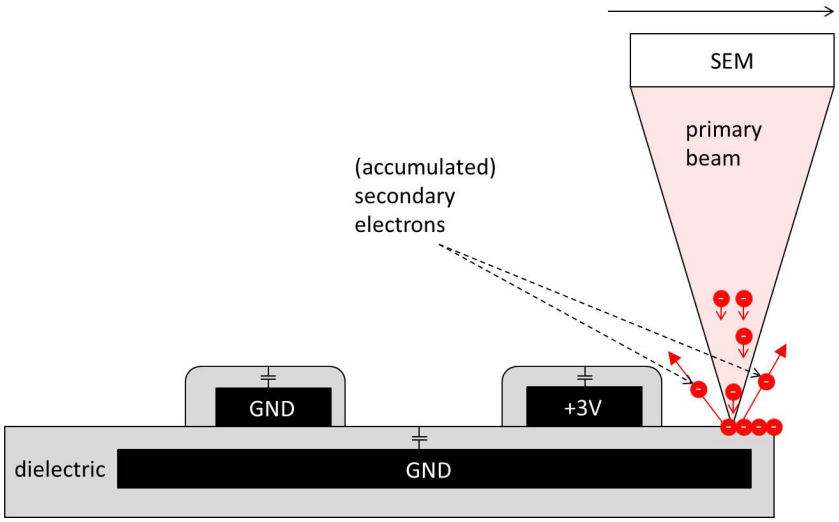
d_E are the extraction grid potential and distance, while d_p and W represent the buried line depth and width, respectively[18]:

$$\frac{U_S}{U_p} = f(U_E, d_E, d_p, W) \tag{3.1}$$

The CCVC phenomena is made visible with a SEM through dynamic changes. Therefore the CCVC is separated into two phases shown in Figure 3.2.



(a) Charging phase



(b) Read-out phase

Figure 3.2: Capacitive Coupled Voltage Contrast with its two phases. First charging phase accumaltes charges. After a clock cycle the metal changed and the accumulated charge is release when hit by an electron.

The first phase charges up the ICs surface with the electron-beam with location-depending accumulated charges (due to U_S), coupled to the underlying electrical potentials (U_p). In the

second phase, the accumulated electrons are hit out as secondary electrons. Compared to the normal structures, are more electrons detected, 'brightening' the structure. We can utilize this phenomenon when the electrical potential of the top layers is data-dependent. Please note that this is not always the case as modern CMOS processes route the VCC or GND signal through the top metal layer. The data dependency leaks further information in a SC that are meant to be kept secret. This has already been seen as a theoretical threat in [30]. We show a practical attack and utilize the CCVC as a SC, not considered in the SC community so far[2].

Furthermore we emphasize that the authors of [122, 123] show a possibility for backside CCVC or E-beam probing (EBP). This imposes a big threat for IC vendors and designs if backside CCVC is scalable to big areas like the shown frontside approach. Backside CCVC has the potential to become one of the most threatening SCs, as there are almost no IC backside protections in today's IC structures. In this chapter we show a POC for the frontside CCVC that is extended to the backside in Section 3.6. We refer to frontside CCVC throughout the rest of this chapter, if not stated otherwise.

Note that the DUT needs to be depackaged and we require the passivation layer, which classifies the CCVC as a common semi-invasive approach. If the attacker is able to remove the passivation with the DUT still operational, other DVC s are possible as well. Therefore we will stick to the term of DVC throughout this chapter, rather than CCVC. Figure 3.2 shows the electrical properties of the CCVC, separated in charging phase in Figure 3.2a and read-out phase in Figure 3.2b.

We described in Section 3.3.2 the possibility to distinguish between high and low voltages on the surface from the brightness in the SEM image. With carefully selected SEM parameters, we can see dynamic changes during the clock transition for a short time. By optimizing the parameters, we were even able to observe changes in the second metal layer[3], easily distinguishable as 90° rotated wires.

As the CCVC is built from two phases and especially the first phase needs some time to charge the surface, the clock speed of the DUT has to be very slow. With an external clock, this can be done in a trivial manner. In more complex scenarios, clock stretching [28], invasive mechanical probing or even EM-based attacks on ring oscillators [20] could be feasible. By reducing the size of the ROI in the SEM a clock speed of some Hz to some kHz might be possible, as only this small region is charged and read-out. Other DVC techniques solve this problem by introducing a pulse gate in stroboscopic SEMs [148]. Academic publications show scans within GHz clock speed range[149], while commercial EBP products can be found with similar capabilities[123].

3.4 Voltage Contrast Analysis

In this section we describe a DVC analysis for successfully locating the AES circuit of our DUT. The DUT is a decapsulated XMEGA32A4U, an 8-bit uC with a dedicated AES hardware unit.

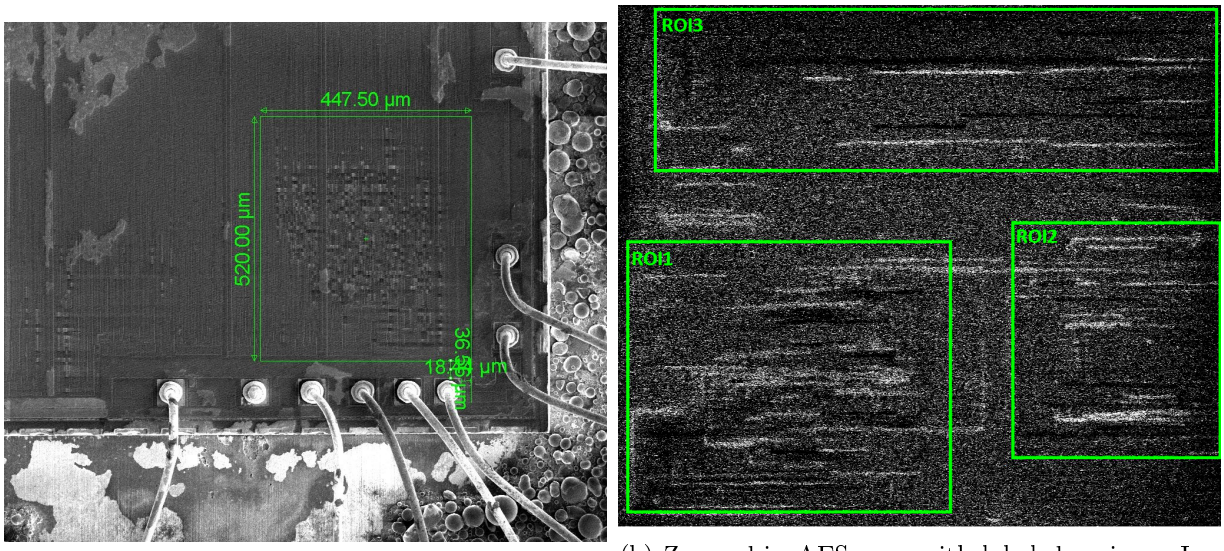
[2] To the best of our knowledge
[3] layers from top to bottom

The AES-128 core needs 376 clock cycles to en- or decrypt blocks of 16 bytes, with the option to `xor` the result once more for different AES modes [16].

The user can read the last round key from the key register. After each block encryption, the key has to be set again. The AES hardware is driven by the peripheral clock, which can be fully controlled externally or can be set to a multiple of the internal generated CPU clock.

Locating the AES Circuit

As a first test, we tried to identify the AES circuit by looping the AES encryption with unknown data at 2 MHz, while performing a DVC in the SEM. The CPU is in sleep mode during the encryption. The electron beam accelerating voltage is set to 1 keV and the beam scanning time is set to repeat as fast as possible. We achieved best results with a Through the Lens Detector (TLD). Figure 3.3 shows the result of the DVC.



(a) AES area.

(b) Zoomed-in AES area with labeled regions. Image is saturated for improved visiblility.

Figure 3.3: AES located with DVC. The regions are flickering like noise.

A high activity in the bottom right corner indicates a repetitive computation, assumed to be the AES circuit. To verify our assumption we run a normal CPU program without using the AES core and observed the ROI. As it did not show any activity during the verification run, we concluded that the ROI is indeed the AES.

As shown in Figure 3.3 we were already able to identify the AES circuit by simply running the algorithm and observing the top metal layer using the SEM. We note that this step is considerably simpler and faster compared to other approaches such as EM cartography, thermal imaging or photon emission discussed in Section 3.2. Also, the EM trace acquisition is often non-trivial. Subsequently, a reverse engineer can focus his attacks on the ROI.

3.5 Voltage Contrast Side Channel Analysis (VCSCA)

This section is the main contribution of this chapter. We describe a Side-Channel Analysis (SCA) with the DVC explained in Section 3.3.3. We use the VC as a side channel and perform a SCA to retrieve additional information of the AES circuit positions. Additionally, we recover the full AES key in 2 SCAs explained in Section 3.5.4 and Section 3.5.5. However, the attack has a significant potential as a general-purpose tool for extracting data and performing hardware reverse engineering against unknown circuits. This tool has a big potential, even for modern CMOS processes, if we include backside CCVC shown in [122], [123] and Section 3.6.

Before explaining the VCSCA in more detail, we want to point out, that we performed an optional EM-based collision-attack in advance. This additional SCA was done to synchronize the retrieved byte-order and timing information with ROI 1-3 in Figure 3.3b. This allowed us to identify ROI 1 as the `addroundkey` subroutine. After this section, this information became obsolete, as the DUT is vulnerable to the more powerful introduced VCSCA. Nevertheless, the combination with another SCA is a general-purpose approach to further reduce the ROI, even if the VCSCA is not applicable. We describe the setup and attack of the VCSCA in more detail during this section.

3.5.1 Obtaining Voltage Contrast Traces

A straightforward AES encryption program for the XMEGA has been written. It receives a 16 byte key and plaintext over USART and encrypts an AES-128 data block. Just before the AES is starting, the clock is set to react to an external pin and the main CPU is configured to enter sleep-mode.

In Section 3.3.3 we explained that the accumulated charges from the charging-phase disappear quickly after the clock transition. Hence, we have to time a single picture very accurately. To circumvent this problem, we decided to start recording a movie using the SEM software and cover multiple clock cycles in one recording. During the VCSCA the clock has been set to 3 Hz. The SEM parameters are the same as described in Section 3.4. The final setup can be seen in Figure 3.4.

Figure 3.4 shows the setup of the VCSCA. This rather complex setup is needed as the DVC needs to be synchronized with the external clock. The dynamic changes appear only for a brief moment, which led us to start recording a movie. For synchronizing the DUT clock with the recording, an uC (3) is set up to simulate a keyboard to start the recording of video traces within the SEM -control Software (2). The clock is set to 3 Hz as this is the optimized speed to see each DVC change on the surface, without overlapping charging effects from the clock cycle before. This can be seen in Figure 3.5. About every 4-5th frame in the recorded video is a 'clock-frame'. They have visible DVC changes and minimal charges from the previous clock cycle. The PC (4) is used to generate, send and validate the plaintext, key and received ciphertext. 300 video traces with 200 frames each were acquired for the VCSCA. Each frame has the image resolution of 1024 x 885 pixels. Plaintext and keys are chosen randomly, but are known.

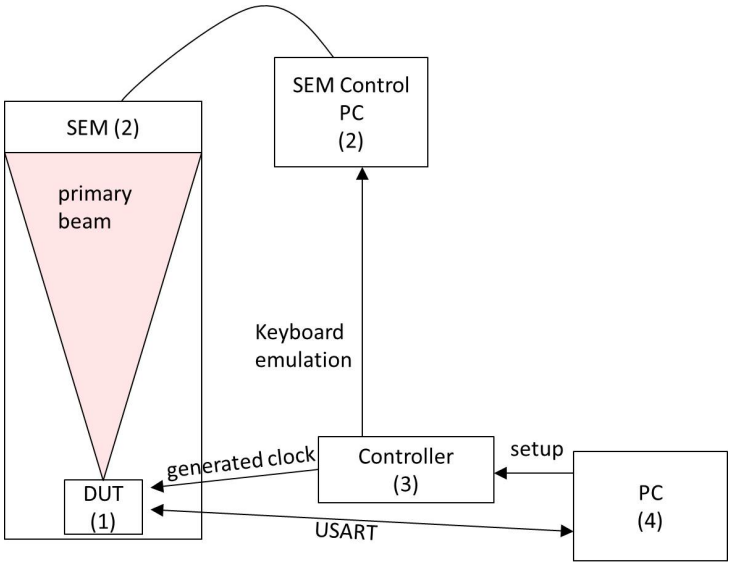


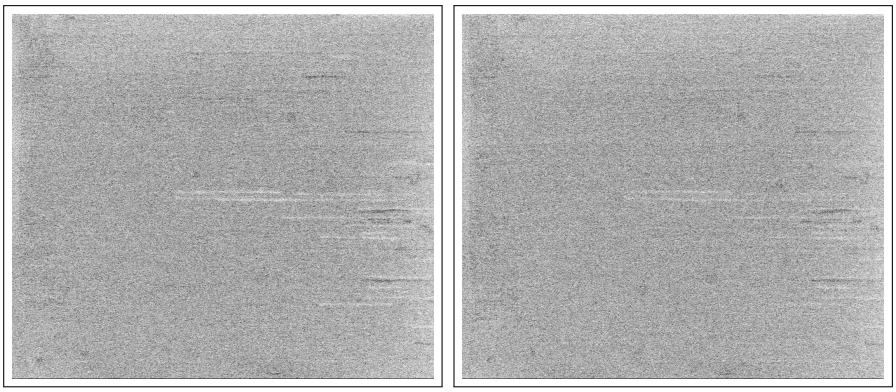
Figure 3.4: The setup for the VCSCA. We run the DUT within the vacuum chamber while externally controlling the clock frequency.

3.5.2 Locating AES Bit wires in a VCSCA

In this section, we determine the Pearson correlation coefficient between all pixels and the emulated internal AES bit values. The AES is emulated in software with known key and plaintext. Therefore we know every intermediate value but concentrate on the first AES round. The correlation is done on every pixel in every frame extracted from the (video-)traces. Overall this makes $1024 \times 885 \times 300$ values for each frame and hypothesis to calculate the Pearson correlation from.

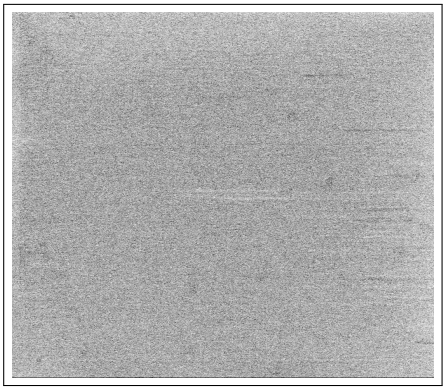
Note that we are using the absolute bitvalue (Hamming weight) of single bits and know the key, plaintext and respective processing order from an optional SCA collision Attack. This reduces the computation time significantly, as we know in which frames (clock cycles) the bytes are processed. The CPU overhead for calculating the hypothesis can be neglected. This step can further be optimized to work on some smaller regions and selective pixels if necessary. Please note that we do not try to optimize our trace number or calculation time. Other possible correlation-based approaches could lead to better results but are not the focus of this thesis. The result of the differential VCSCA can be seen in the correlation image in Figure 3.6.

Figure 3.7 shows the correlation images of further bits after different AES subroutines (`addroundkey` and `subbytes`). Each has different peak-locations, showing the position of the processed bits. Images labeled with 'inverted' invert the sign of the Pearson correlation. Each image cuts negative correlations to 0. With this we get two different images for each bit processed that show logically inverted signals and their locations as well. The corresponding hypothesis and respective clock cycles are listed in the subcaption. A high correlation for `addroundkey`, `subbytes` and the plaintext was found in respective 16 consecutive clock cycles. Interestingly we found the plaintext bits on the very same spots as the `addroundkey`, indicating a common load/store unit or bus structure within the AES.

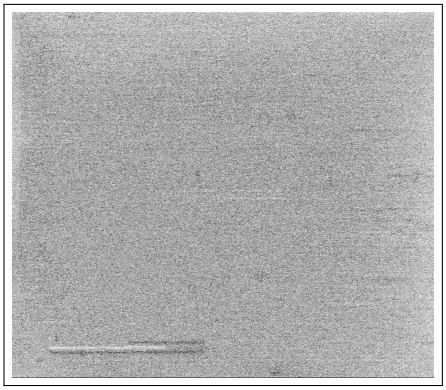


(a) frame x

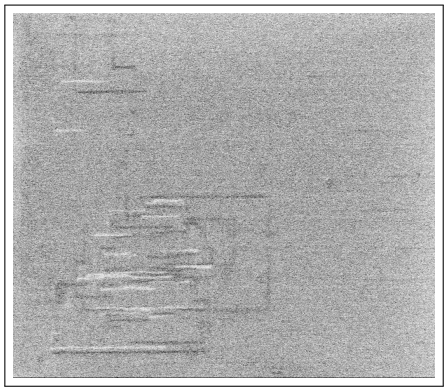
(b) frame x+1



(c) frame x+2

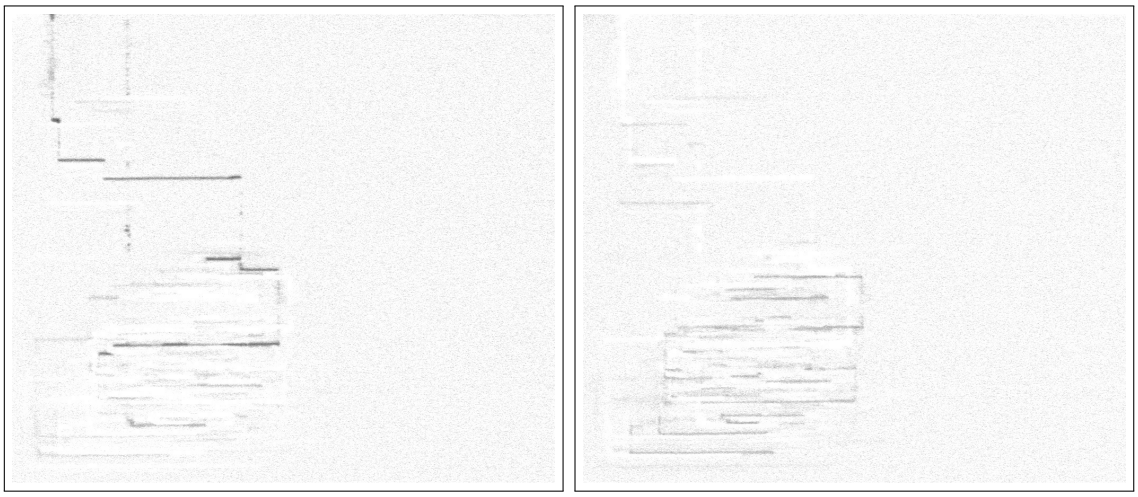


(d) frame x+3



(e) frame x+4

Figure 3.5: Consecutive frames within one trace. A clock transition takes place, while the SEM scans the last third of frame x+3. The previous clock effect fades out (a)-(d). The colors are inverted for improved visualisation.



(a) highlighted bit in a VCSCA (b) target bit inverted

Figure 3.6: Results of the correlation based VCSCA of the 8th addroundkey-bit in clock cycle 42. The colors are inverted for improved visualisation.

This information immediately allows an attacker to retrieve unknown keys in a template based VCSCA approach. This attack is done in Section 3.5.4 to recover the full AES key.

Before we introduce a template based key extraction we would like to emphasize the fact that the correlation images provide valuable information for a reverse engineer or sophisticated attacker. We know the location and meaning of selected wires in the highest metal layers. This reveals the location of AES calculations on gate-level when the wires are tracked into the FEOL(polysilicon layer). The attacker is able to interpret neighboring and connected signals immediately. This might reveal further weaknesses or even whole Intellectual Property (IP) cores.

Furthermore, it is also possible to apply other approaches easily, as the locations of the AES bits in top metal layers are known. Mechanical probing or fault attacks are two examples to retrieve or alter intermediate AES bits. Figure 3.8 shows cropped images of the 2 top metal layers from the XMEGA microcontroller scanned with a SEM, overlayed with the extrapolated correlation image of `addroundkey` bit 2. The correlation image is 50% transparent to visualize the manual mapping process.

In Figure 3.8 we demonstrate that we are capable of identifying the tracked wires of bit 2 in the first two metal layers. Continuing this, we would be able to pinpoint the location of the origin in the polysilicon and the next 'processing steps' after the `addroundkey` subroutine. Nevertheless, this is out of the scope for this specific attack. It is noteworthy that the two marked wires from the top layer are connected in the second layer. This is a good indication that we hit the right wires from our results.

(a) ark bit1

(b) ark bit2

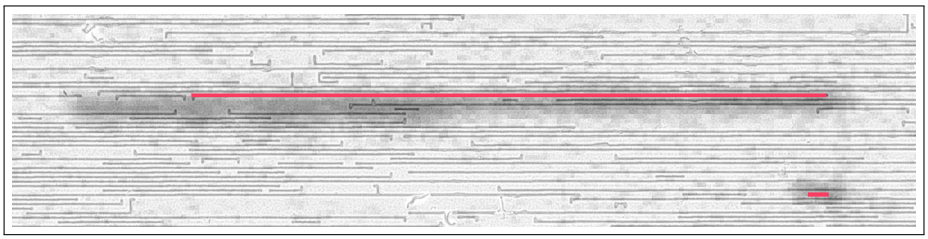
(c) inv ark bit1

(d) inv ark bit2

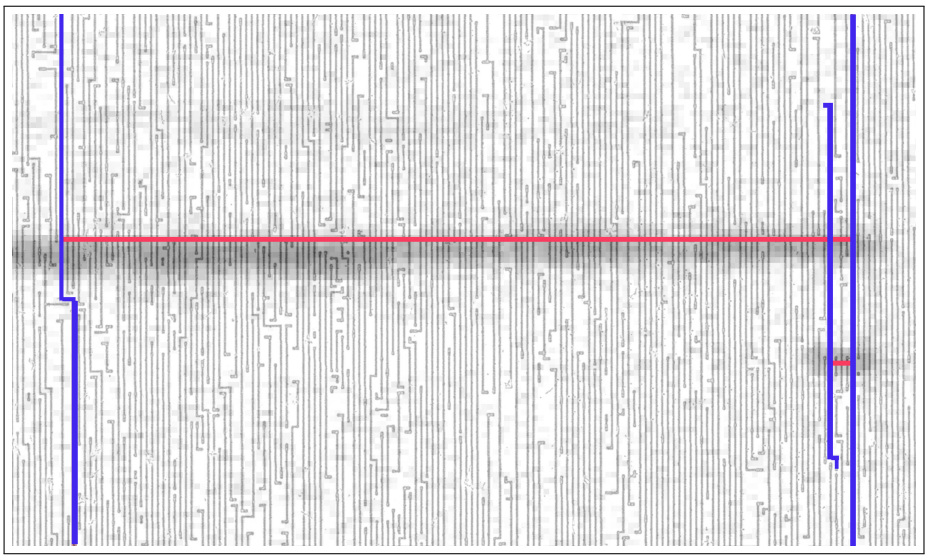
(e) sub bit1

(f) sub bit2

Figure 3.7: Different correlation images found in clock cycle 42; addroundkey, inverted addroundkey and subbytes. The colors are inverted for improved visualisation.



(a) Top metal layer



(b) Metal layer beneath the top metal layer

Figure 3.8: Marked wire of bit 2 of addroundkey in the two top layers. The black “cloud" is the extrapolated correlation image in this ROI. The colors are inverted for improved visualisation.

3.5.3 Extracting additional netlist information

So far we analyzed bits within the AES circuit that are supposed to be part of the AES calculation. We did not look into the possibilities of how the `subbytes` routine or other subroutines are built. Therefore we provide a SCARE like approach in this section.

We applied every $2 \rightarrow 1$ function from the 8 `addroundkey`-bits possible and used the result as a new hypothesis. Each possible $2 \rightarrow 1$ function is given in Table 3.1 from [68]. The results of the 2bit-function hypotheses revealed additional circuit operations not known so far. For example, we did find a correlation of a `xor` between Bit 3 and Bit 4 of the `addroundkey` bits. This allows us to build basic netlist operations and we are able to verify an extracted netlist (by usual means) with these findings or counter hardware obfuscation techniques like the one introduced in [110]. Some hardware protection might even play in our hands by routing sensitive wires through multiple layers, including the top metal [33].

Following this approach we are able to (partly) reverse engineer whole subroutines, without invasive methods. This is not in the scope of this chapter and can be done in future work. This

Index of f	f(B,A)				Boolean equation	Name
	$f(1,1)$	$f(1,0)$	$f(0,1)$	$f(0,0)$		
0	0	0	0	0	0	Zero
1	0	0	0	1	$\overline{B + A}$	NOR2
2	0	0	1	0	$\overline{B} \cdot A$	AND2B
3	0	0	1	1	$\overline{B}$	NOTB
4	0	1	0	0	$B \cdot \overline{A}$	AND2A
5	0	1	0	1	$\overline{A}$	NOTA
6	0	1	1	0	$B \oplus A$	XOR2
7	0	1	1	1	$\overline{B \cdot A}$	NAND2
8	1	0	0	0	$B \cdot A$	AND2
9	1	0	0	1	$\overline{B \oplus A}$	XNOR2
10	1	0	1	0	A	A
11	1	0	1	1	$\overline{B} + A$	OR2B
12	1	1	0	0	B	B
13	1	1	0	1	$B + \overline{A}$	OR2A
14	1	1	1	0	$B + A$	OR2
15	1	1	1	1	1	One

Table 3.1: $2 \rightarrow 1$ functions. The output bits are used as a hypothesis in the Pearson correlation with intermediate AES bits. Once a high correlation is found, a respective connected cell in the front-end can be identified.

small POC shows the great potential for the VC side channel in SCARE like approaches.
In the following sections, we use the VC in more common SCAs, to recover the AES key with our current setup. We execute a template-attack and independently another SSCA on a single trace in a no-plaintext, no-ciphertext, and no-key attack. In both cases, we retrieve the AES key successfully.

3.5.4 Template Attack with VCSCA

The setup for the key-recovering template attack is the same as described in Section 3.5.1, with the difference of choosing a constant (assumed unknown) key for all the traces. 250 Traces are acquired and random plaintexts are AES-128 encrypted by the DUT.
The 'templates' are the correlation images generated in Section 3.5.2. The idea is to correlate the frames that process a specific byte and correlate the resulting `addroundkey` bit with a single key bit hypothesis. The resulting correlation image is either the 'normal' or the 'inverted' correlation image e.g. of bit 8 shown in Figure 3.6.

To explain the process, we give a short example of the 8th bit of byte 16 of the i-th trace ($p_{i\_16\_8}$). As we know the that the 16th byte is processed in the 42nd frame of each trace, we extract this frame from every trace. Let us assume that our hypothesis of the 8th keybit of byte 16 is "1". We correlate each pixel of these frames with the `addroundkey` hypothesis which is calculated ($p_{i\_16\_8}$ `xor` hypothesis). This results in a correlation image that is either close to Figure 3.6a or Figure 3.6b. If the assumption is correct, we will get a correlating image close to

Figure 3.6a. Otherwise, the hypothesis is wrong and the 8th bit is “0". Repeating this process, every keybit can be recovered. Taking 250 traces is an estimated value to make sure that the attack succeeds, as we need to extract the right clockframes of the DVC video and to get a good average over noisy images. We verified that the attack is feasible with fewer traces.

3.5.5 Simple VCSCA

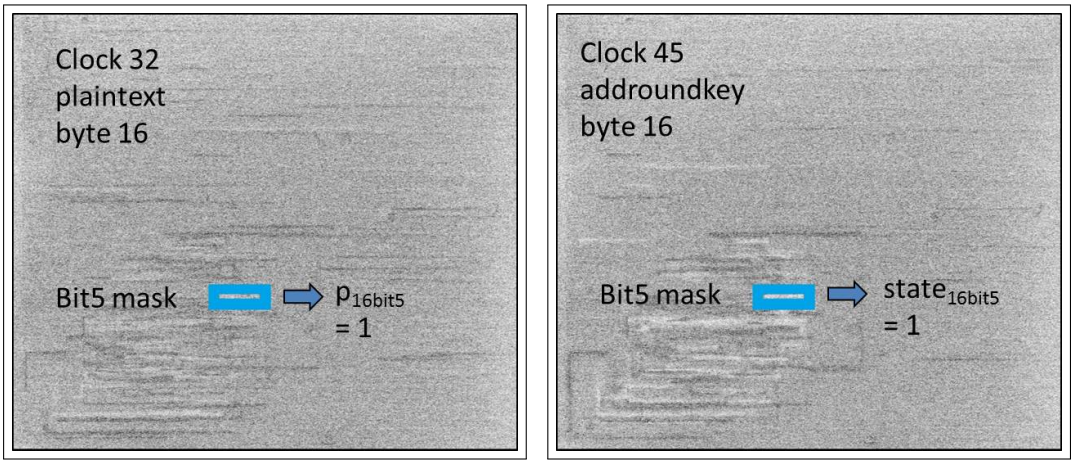
Realizing that the XMEGA reuses the same circuit for every byte sequentially, the bit locations are the same for every round and byte within the AES. Therefore, we aim to find plain or key bytes directly being loaded or processed during the AES setup. Interestingly we found the plaintext being loaded 13 clock cycles before the `addroundkey` function on the same bit locations as the `addroundkey` bit. This indicates a common load/store unit or a bus architecture. In this section, we assume not knowing the key once more. Knowing that the plaintext is being processed right before the `addroundkey` allows an attacker a no-plaintext, no-ciphertext **and** no-key SSCA against the XMEGA AES engine. The aim of this section is, therefore, a POC of the simple VCSCA with a single trace and 1 byte.

Section 3.5.2 already revealed the location and timing of individual `addroundkey` bits being processed. These bits have a key dependence through the `xor` with the plaintext, which can be read-out 13 clock cycles before. Therefore, this attack first recovers the plaintext bit and secondly reads-out the processed `addroundkey` bit in a SSCA. The recovered bits are `xored` to get the corresponding keybit as we only target the first round. This is repeated for 8 bits within byte 16 in a single trace. Figure 3.9 shows the wire positions for 2 bits that we used for recognition. The wire positions are chosen from multiple options, as they are reliable measured by the DVC and are easily recognizable.

(a) Bit 2 (b) Bit 5

Figure 3.9: Correlation Image of keybits within clock cycle 48. The colors are inverted for improved visualisation.

In Figure 3.10 we demonstrate the simple VCSCA based on Bit 5. Other bits can be read-out in a similar manner.



(a) Clock 32 plaintext (b) Clock 45 addroundkey

Figure 3.10: Extracted states of byte 16 bit 5 within one trace. We know that bit 5 of byte 16 runs through this wire at clock cycle 32. Since the wire within the mask is bright, the bit is set. (The colors are inverted for improved visualisation).

A direct `XOR` of both values, reveals the 5th keybit of byte 16. The keybit correctness was verified for every bit in Byte 16. We discovered that is not easy to find a trace that shows all bits recognizable at once, as the DVC images depend on the previous top-metal layer voltage and the exact beam position during the clock transition. We verified that is is possible to acquire at least one key-byte with one single trace. We did not look for further keybits since the recognition was done manually. It is possible to automate this process in future work when more traces are used to work on averaged images.

3.6 Backside Voltage Contrast Analysis

In this section, we demonstrate the capability to acquire backside VC traces. Related work in[122] and [123] have shown that backside EBP is possible with certain preparation effort. Their work is aimed at the FA community and shows the feasibility for a single location. Our aim is to extend the preparation and show that a hardware security threat is present.

In an abstract overview, we thin the backside of a DUT after a ROI detection and read the underlying interconnects with the VC. We try to extend the Shallow Trench Isolation (STI) trenching to cover multiple bits at the same time. This allows to apply the introduced VCSCAs from previous sections. To show the feasibility of such an attack, we acquire VC traces and argue that the attack is feasible.

As we require an preparation from the FIB we first need to pinpoint our ROI, see Section 3.6.1. In the following, we prepare the ROI with an CNC drill to remove the bulk silicon. Then, the FIB is used to remove silicon in a controlled, limited area process, see Section 3.6.2. After the preparation we obtain backside traces in Section 3.6.3. They are comparable to the frontside Traces in Section 3.5.1 and can be used for the same approaches from Section 3.5.3, Section 3.5.4 and Section 3.5.5.

3.6.1 ROI identification

Several methods for ROI detection have been discussed in Section 3.2. Once the ROI is located, we can prepare the backside of the IC with the ROI in mind. With current FIB capabilities, the ROI can be roughly $100\mu m$ by 100 μm. Note, that the area can be extended significantly with state-of-the-art plasma FIB or similar Broad Ion Beam (BIB) machines. Here, the challenge is to remove silicon uniformly which is the limiting factor for current FIB optics.

In the following, we can skip the process of ROI pinpointing, as we know where the AES is located from the previous sections. Hence, we arbitrarily chose a $100\mu m$ by 100 μm area directly underneath the AES location. The chosen ROI can be seen in Figure A.1 in Appendix A.1 .

3.6.2 Preparation

For the backside attack, we need to remove several μm silicon from the bulk (usually $>$150 μm) in multiple steps. First, the chip is mechanically polished to reduce the biggest part of the Si in a coarse way. The silicon is mostly used to stabilize the whole IC and thus, not critical while the chip is used. This process thins the bulk as close as possible to the backside without damaging the chip and bond wires. Here, 10 to $15\mu m$ Remaining Silicon Thickness (RST) are feasible without damages to the structures. Afterward, the last two steps are done in a FIB or BIB/Ion Mill to slowly sputter the remaining silicon without damaging active areas. We sketch the three steps in Figure 3.11.

The XMEGA32A4U is backside thinned with a dedicated CNC to around 15 μm RST. Please note that this preparation step is very challenging for RST$<5\mu m$, as warp packages, heat, and mechanical limitations. The dedicated equipment usually comes with a high price tag.

In the following, the sample is further thinned as illustrated in Figure 3.11. By thinning the remaining silicon with the FIB down to STI level, we are able to see the p- and n-active regions with VC in the FIB live view. This can be seen in the traces in Figure 3.13. This means that once a FIB is accessible, the operator does not need to be an expert when the sample is monitored during the process. The process duration is depending on the RST and the FIB parameters within some hours. Please note that if the processing time is important, different gases and plasma FIB can be used.

Once the ROI is evenly at STI level, we carefully trench the STI only (which can be seen in Figure 3.13). We can see the active regions in the FIB to navigate and cut into the STI. Depending on the remaining SIO_2 underneath the gates, we achieve as similar CCVC or PVC effect as described in Section 3.3.1. The STI trenches are placed in straight lines and are usually very narrow. We trench a complete row within minutes. This allows seeing the gates on the poly-silicon layer and their VC changes, see Figure 3.13.

In the case of complete STI removal, the state is not volatile as we describe a static AVC. Nevertheless, this process cannot be maintained indefinitely as the structures are hit by electrons. This influences the charge and state of the wires and gates significantly [123]. In fact after several minutes of running the default AES engine within the SEM, we encountered different ciphertext outputs due to the electron charges. An example of the trenched STI over multiple interconnects can be seen in Figure 3.12.

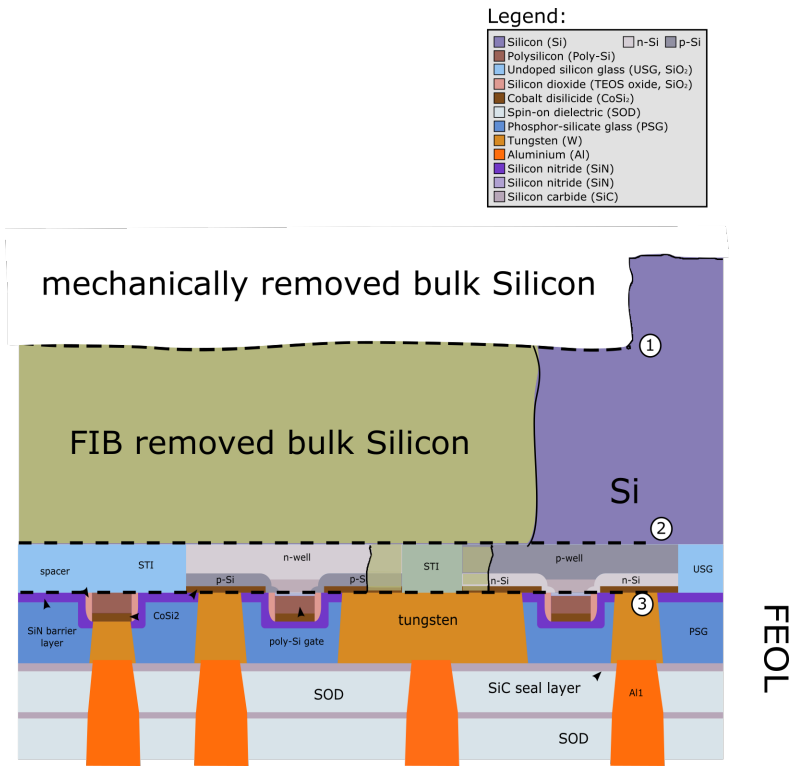


Figure 3.11: The IC is thinned from the backside. In the first step he is mechanically polished down to approximately 15 μm. The second step is done in the FIB until the p- and n-wells can be seen, see Figure 3.13. The last step trenches the isolation between the gates, see Figure 3.12.

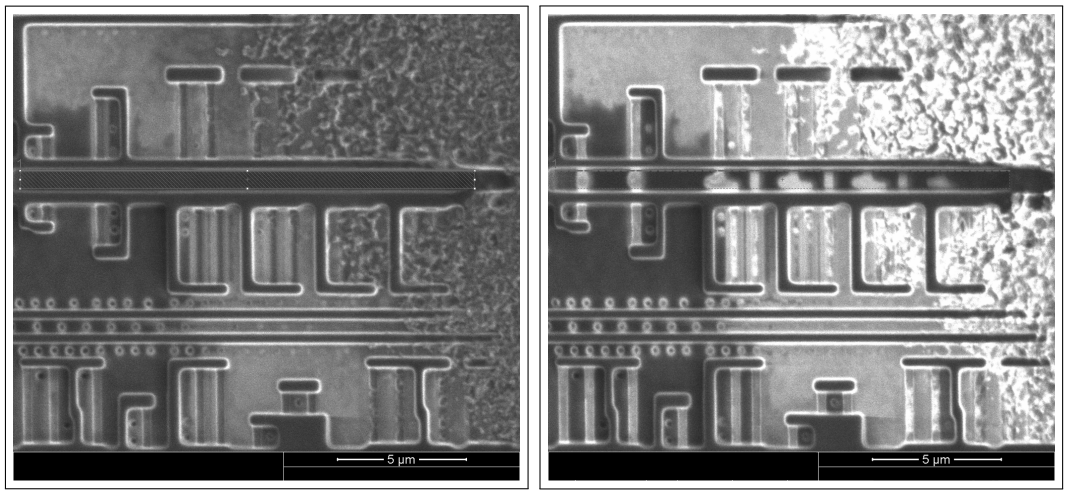
3.6.3 Backside Traces

After preparation we run the AES hardware engine with an external, controlled clock as in Section 3.5.1. We are able to see VC changes in a very narrow trench of the DUT. Hence, we obtain VC traces from the ultra-thinned backside approach, see Figure 3.13.

With the acquired traces the the techniques from Section 3.5.2 and Section 3.5.4 can be re-applied. Nevertheless they did not match any known AES bits. We think, the arbitrarily chosen ROI did not have any intermediate values.

3.7 Discussion

In this section we want to state the limitations and possibilities of the VCSCA. The VCSCA is related to known EBP techniques from the FA community, but extended for hardware reverse engineering and SCAs. We note that current ROI detection techniques, see Section 3.2, are versatile and suitable for ROI detection in modern technology nodes. Once a ROI has been determined, the front and backside approaches described in this chapter can be applied.



(a) Backside ROI without any Si (broken) (b) Backside ROI removed STI

Figure 3.12: STI trenching in a small area on a sample. The STI can be cut in a straight line reducing preparation time and steps.Within the trench the polysilicon interconnects can be seen and read by a voltage contrast. To the right we see a thin layer of remaining silicon.

Frontside Limitation With the advances in the semiconductor industry the number of layers have steadily increased to nowadays more than 10 layers with the uppermost layers being a GND and VCC distribution network. Only small uC, and ASICs, are build in old technologies using all of the 4 to 6 metal layers for routing. Hence, the frontside VCSCA is applicable to small uC like our XMEGA32A4, but has less chance to succeed in modern ICs. Meshed high security chips and modern sub 20 nm technologies are not vulnerable to frontside VCSCA. The uppermost grounded layer acts as a shield against other (semi-invasive) attacks, such as EBP, frontside photon emission, EM SCA, ... Even FA techniques with perfect knowledge of the IC layout face the problem to contact buried signals in a 10 layer stack for debugging purposes.

Backside Limitation By opening the IC from the backside, we tackle prior shortcomings and ICs security mechanism, e.g. meshes and shields. Backside preparation techniques are researched with great interest from the FA community, hence vendors usually provide dedicated equipment that thins down to $5\mu m$ RST [114, 39]. Mainly the following ultra thinning to STI level[123, 39] requires advanced preparation equipment like the FIB, increasing the resources for this approach. Please note that a FIB can be rented or outsourced in third-party services. Even though the FIB work requires expertise, the end-point is remarkable. In the end, the STI trenching (step 3 from Figure 3.11) is not trivial and requires experience. Nevertheless, we conclude that any adversary with access to a FIB is able to prepare the target with moderate costs. From a security point of view we would like to highlight that the authors do not know of any (practicable) countermeasure against backside deprocessing.

During this work, we required multiple samples during the last preparation step and, worth noting, the VC with the electron beam has an influence on the processed internal data. A some point in time, the wires under inspection are charged due to the electrons hitting its surface.

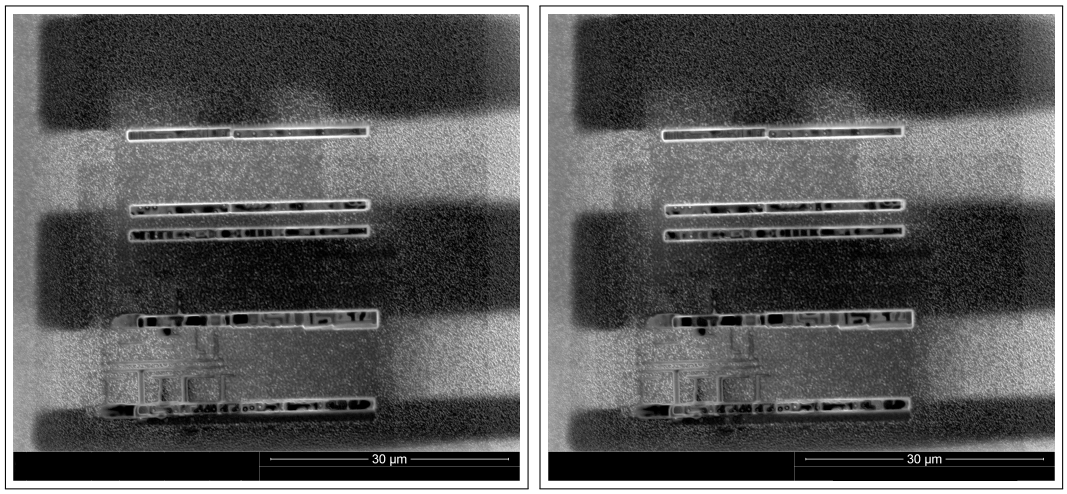


Figure 3.13: 2 frames from the backside VC Traces. The bright and dark yield are the dotted active areas. The trenches hit the STI to see the metal and poly lines in a VC. One trench missed the STI due to a miscalculation, rendering this sample broken. Changes within the trenches are logical high or low interconnects and gates.

This induced erroneous behavior during inspection and requires a good preparation and VC parameter tuning in the SEM or FIB. In the latter case, the sample is further sputtered and thus destroyed by hitting ions. To understand the limits and capabilities of backside VC, important parameters, such as electron acceleration, detector technique, process materials and electron beam intensity per area are important parameters require further research [123]. Each sample requires fine-tuning to find the fine line between erroneous behavior and an undisturbed VC analysis.

Sequential architectures repetitively working on parts of data are especially vulnerable, as consecutive parts(e.g bytes) are processed at the same position that can be probed at different times. Note, that in the case of missing crucial keybit locations, key material can be retrieved with differential cryptanalysis using intermediate bits [29]. With the gained knowledge a reverse engineer has a perfect start to mark and follow the signals in a reverse engineered layout, see Chapter 2. Furthermore, additional hardware reverse engineering explained in Section 3.5.3 and advanced SAT-based reverse engineering for gate-level schematics [82] can be applied in future work.

3.8 Conclusion

In this chapter we revise an approach to pinpoint the ROI for IC s with VC images. This reduces the complexity of hardware reverse engineers by gaining apriori knowledge of the location of security-relevant ROI and allows to perform EM-based SCA with better signal-to-noise ratio. Furthermore this enables more advanced SCA s like inter-gate leakages discussed in [137] and reduces the exhaustive search for fault attacks.

The shown approach is at least as fast and easy to comparable approaches like EM cartography, thermal imaging and photon emission. The only required tool is a SEM, well distributed in universities and institutes due to its application in many academic fields. A SEM can be bought second hand for under 10k USD. Specialized commercial EBPs for high speed measurements are also available.

We use the VC as a side channel that exploits the capacitive coupling effect of top metal layers through the covering passivation. We are able to see voltage alterations of the IC s surface in a SEM, revealing secret information through top metal-wire voltage changes. A POC with a XMEGA microcontroller to locate and identify top metal layers holding intermediate AES bits is given. Any sophisticated attacker can track the wires to the polysilicon layer, revealing data flip-flops and memory structures of the AES. Furthermore we use the VC in multiple SCA approaches to recover the full AES key.

For the VCSCA template-attack, less then 100 traces are enough to reveal the key, while the simple VCSCA is performed in a no-plaintext, no-ciphertext and no-key attack scenario. Additionally we show a SCARE approach to recover further netlist information, that can be used to reverse engineer hardware circuits in a non-invasive way. The gained information can be used to partly verify an extracted netlist or to counter simple hardware obfuscation techniques.

Finally, the backside VC or EBP preparation initially shown in [122] and [123] can be expanded to area sizes relevant for the backside VCSCA as we have shown in Section 3.6. The VCSCA is a high hardware security threat, as the shown front side VC approaches are also applicable to the unprotected IC backside.

Chapter 4

Security Implications of Intentional Capacitive Crosstalk

With advances in shrinking process technology sizes, parasitic effects of closely routed adjacent wires so-called crosstalk are still a relevant problem in practice since they directly influence performance and functionality. Even though there is a solid understanding of parasitic effects in genuine hardware designs, the security implications of such undesired effects have been scarcely investigated. This chapter is based on a publication in the IEEE TIFS journal [84].

Contents of this Chapter

Contribution: *This chapter is a joint work with Omar Awad and Marc Fyrbiak under my supervision.*

4.1 Introduction

Since the early days of IC design and manufacturing, parasitic effects, such as crosstalk, have been a major issue for the digital electronics industry [67]. These effects occur due to densely routed interconnects, thus posing an important challenge for advancing technology sizes as parasitic effects impair functionality and reduce performance (e.g., by lowering bus frequencies). To overcome such physical effects, hardware designers have developed strategies to analyze placed-and-routed hardware layouts prior to manufacturing for parasitic capacitances, resistances and inductances [155]. This step creates a precise analog circuit model so that circuit simulations can

be investigated for undesired physical effects. Several mitigations have been proposed to combat parasitic effects such as increasing the space between wires and the use of data encodings [152].

As modern IC design and fabrication processes have been moved offshore, increasing distrust among participating stakeholders, the need to ensure the trustworthiness of manufactured ICs has risen dramatically [26]. Malicious circuity *a.k.a.* hardware Trojans, can invalidate sophisticated security measures or deny an IC's service, thus posing a significant threat to today's military systems, financial infrastructure, transportation systems, and safety and household appliances [143]. Since significant resources are required to analyze and detect hardware Trojans in manufactured ICs, (e.g., visual inspection), only a handful of institutions can address such concerns [143].

To make matters worse, several scientific works have demonstrated parametric design strategies [66] based on physical parasitics in IC manufacturing that exhibit close to zero logic overhead. These approaches are hard to detect with non-invasive or semi-invasive techniques. As Trojan design advances further, only invasive hardware reverse engineering strategies will be able to counter such Trojans, particularly when no golden model is present or the golden model must be verified [145, 143]. As a result, the search for countermeasures has become of increasing interest; to (1) identify Trojan circuits early [26], (2) secure the IC design [153, 52, 4], and (3) secure the production chain [74, 15].

Goals and Contributions. In this paper, we focus on parametric hardware Trojans based on parasitic effects. Our goal is to demonstrate that crosstalk-based effects can be reliably exploited to realize adversary-controllable faults in digital logic implemented only by rerouting existing wiring resources. The approach has a zero-gate overhead. To this end, we present a general Trojan design methodology for crosstalk Trojans and confirm its devastating consequences in two security-relevant case studies. Finally, we present a novel countermeasure to detect such Trojans by enhancing state-of-the-art visual inspection techniques.

In summary, our main contributions are:

- **Generic Crosstalk Hardware Trojan.** To the best of our knowledge, we present the first generic building block for parametric hardware Trojans based on capacitive crosstalk. We show that almost any victim wire can be targeted by an adversary so that the DRC is not violated. We also illustrate that such Trojans are stealthy and hard to detect with currently employed visual inspection techniques and hardware Trojan detection strategies.

- **Security Implications.** We assess the threat and capabilities of the crosstalk Trojan using two offensive case studies: (1) we subvert the security of an AES IP core to leak the secret key and (2) we perform a privilege escalation in an OR1200 CPU which is capable of running a modern Operating System (OS) and executing arbitrary code with elevated access rights. Both case studies are built using gates from the $45nm$ NanGate FreePDK45 library. Implementation strategies for capacitive crosstalk Trojans are highlighted and future research directions to mitigate their risk are described.

- **Parametric Hardware Trojan Mitigation.** We extend state-of-the-art hardware reverse engineering workflows based on visual detection to identify the characteristic properties of crosstalk hardware Trojans in a semi-automatic matter. Reverse-engineered layout wires are analyzed for parasitic capacitance to highlight potential crosstalk Trojans in the IC design.

4.2 Background and Related Work

We now provide fundamental technical background regarding parametric crosstalk in IC designs. State-of-the-art hardware Trojans (Section 4.2.1) and hardware reverse engineering (Section 4.2.2) techniques are described followed by the assumed threat model (Section 4.2.3).

4.2.1 Hardware Trojans

Hardware Trojans have been of strong interest to the scientific community since an initial report by the DoD [3] in 2005. A hardware Trojan consists of a payload circuit delivering malicious functionality (e.g., the leakage of cryptographic keys or the denial of service) and an optional payload activating trigger circuit (e.g., a counter or side-channel based a factor such as heat or age). For a comprehensive survey of offensive and defensive techniques for hardware Trojans, the interested reader is referred to [26].

Offensive Hardware Trojans Research. Hardware Trojans can be inserted at various stages of design and manufacturing processes (e.g., in RTL or gate-level netlists [83, 94, 109]). Recent scientific works have demonstrated parametric Trojans [158, 22, 66] by exploiting the parametric characteristics of the physical layout. For example, Becker *et al.* [22] manipulated dopant polarity to undermine the security of cryptographic primitives. Yang *et al.* [158] developed analog Trojans as small as a single cell to elevate rights on a modern CPU. Ghandali *et al.* [66] introduced a path delay Trojan that generates a delay fault through the charging or discharging of capacitances by exploiting rare signal paths. Other parametric Trojans use aging-based triggers [128]. The Trojans are inactive after manufacturing, but they become active after a long period of operation. They also can use a reduced supply voltage to trigger a Trojan probabilistically [89]. Further work leaked side channels at Trojan activation [55] and used parametric characteristic to gain side channel information [111].

Defensive Hardware Trojans Research. Defensive research focuses on the detection of hardware Trojans based on diverse characteristics such as triggers, payload features, and physical attributes [143]. To detect Trojan characteristics, side-channel analysis [26] and static and dynamic design analyses [72, 154, 159, 70] have been proposed. If a Trojan is embedded at a malicious foundry, hardware reverse engineering by visual inspection [143] is one of the few, if not the only, approach that has a realistic chance of finding evidence of a hardware Trojan. Several countermeasures aim to protect a design against hardware Trojan insertion through design alteration and modified fabrication processes [153, 52, 74, 15]. While design changes involve hardware obfuscation which can be relatively low cost, fabrication process changes are more economically questionable. For example, splitting manufacturing across multiple foundries results in multiple die pieces which must be stacked. This approach adds non-trivial connection problems known from 3D-integration processes [150].

4.2.2 Chip-level Hardware Reverse Engineering

Deprocessing. Chip-level reverse engineering by visual inspection is an invasive process to extract the functional information of an IC [108, 145]. It consists of alternating phases of delayering and imaging processing of each stacked metal and polysilicon layer(s) of an IC die. Delayering is a combination of wet chemical, mechanical, and plasma processes to optimize the planarity of

the die and SEM contrast for the imaging step. Typically, these steps require expensive equipment and experienced analysts due to shrinking process technology sizes. Detailed deprocessing steps are out of the scope of this paper, but the interested reader is referred to [108, 145].

Image Processing. SEM images can be post-processed to extract wires, vias, and standard library cells in an effort to recombine acquired layer images, thus reconstructing the original design layout (see Figure 4.13 in the Appendix). Typically, image processing involves *cell matching* based on correlation and templates [105]. For example, a cell can be manually selected and manually analyzed to expose its (Boolean) functionality. Then a template is constructed and each repeated instance is identified using correlation or template-based image processing [145].

Gate-level Netlist Extraction. Once all layers are aligned to each other, gates, wires and vias are extracted to acquire a flat gate-level netlist [108]. During this extraction, parasitic information and geometrical wire information are abstracted. The only information available to the reverse engineer is *what* is connected, not *how* it is connected. This approach makes it impossible to locate parametric hardware Trojans since parametric information is lost. Hence, current state-of-the-art IC hardware reverse engineering techniques are insufficient to detect parametric Trojans.

4.2.3 Threat Model

We assume an adversary with the ability to implant a malicious circuit in the layout of the IC design. Such adversarial capabilities occurs in several scenarios: (1) untrusted foundries, (2) malicious third-party IP cores, and (3) malicious insiders and CAD tools. In the case of a malicious foundry, the adversary has to reverse engineer parts of the design to identify relevant gates and signals where the hardware Trojan must be attached [62]. The goal of the adversary is to induce a reliable exploitable fault into the target design via crosstalk effects to realize a subsequent attack. Our threat model is consistent with prior research on hardware Trojans [153, 52, 143].

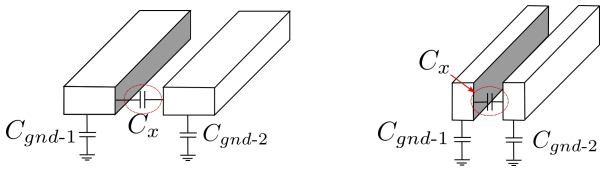
4.3 Crosstalk Trojan Design Methodology

We now provide details of the capacitive crosstalk effect and present our capacitive crosstalk hardware Trojan design methodology that exploits the crosstalk effect between metal lines and wires. The adversary causes a fault by rerouting several aggressor wires to a selected victim wire.

4.3.1 Capacitive Crosstalk

In general, crosstalk is caused by two effects: (1) *inductive crosstalk*, and (2) *capacitive crosstalk*. Inductive crosstalk between two adjacent wires depends on a change in the electric current in one or both wires which generates a changing magnetic field, inducing a current in nearby wires. Capacitive crosstalk between two adjacent wires depends on a change in the voltage in one or both wires which generates a coupling current that generates noise pulses. For modern nanometer technology sizes, both spacings between wires and wire thickness have decreased to reduce parasitic resistance. Since taller and narrower wires are now placed closer to each other, coupling capacitance is the predominant effect for advanced modern technology sizes [63]. Hence,

we neglect inductive crosstalk effects for the remainder of the paper. The interested reader is referred to [155] for more information on inductive crosstalk.



(a) Micrometer Technology(b) Nanometer Technology.

Figure 4.1: In modern processes nanometer technology interconnects are designed thinner to connect more compact logic cells. To counter resistance issues interconnects are manufactured taller which increases the plate capacitor area (marked in gray). While distance is smaller between two interconnects for modern technologies, parasitic capacitance C_x is increased.

Figure 4.1 illustrates these wire geometries. Note that the coupling capacitance C increases for nanometer technology sizes, as the area A (marked in gray) is larger and the distance d is smaller compared to micrometer technology sizes. Equation 4.1 clarifies the connection between area, distance, and the capacitance. Parameter ϵ is the relative static permittivity multiplied by the electric constant ϵ_0[1]. The separation distance d refers to the distance between interconnects and the A is the capacitive area.

$$C = \epsilon \frac{A}{d} \tag{4.1}$$

Based on the increased coupling capacitance, a transition in one wire w_a may generate noise in an adjacent wire w_v due to the coupling current between the two wires. More precisely, the former wire w_a is the *aggressor* while the latter is the *victim*, see Figure 4.2.

In Figure 4.2 the victim wire remains constant while the aggressor wire is switching. During a transition in the aggressor wire, the victim wire voltage is pulled up due to coupling capacitance between the two wires, resulting in a glitch referred to as a *coupling noise pulse*. The resulting noise pulse and noise peak are depicted in Figure 4.3. Note that the peak of the coupling noise pulse in the victim wire occurs when the transition at the aggressor wire is completed. The relevant phase of the noise glitch emerges between t_1 and t_2 when the glitch is above the threshold voltage V_t. If a sequential clock edge signal arrives during this time, an unintended fault in the victim wire is caused. The noise pulse can be accumulated by the switching of multiple aggressor wires at the same time.

$$\Delta V_v = \frac{C_x}{C_{gnd-v} + C_x} \frac{1}{1+k} \Delta V_a \tag{4.2}$$

[1]The electric constant ϵ_0 is defined as $\epsilon_0 = 8.85418782 \cdot 10^{-12} \frac{s^4 A^2}{m^3 kg}$.

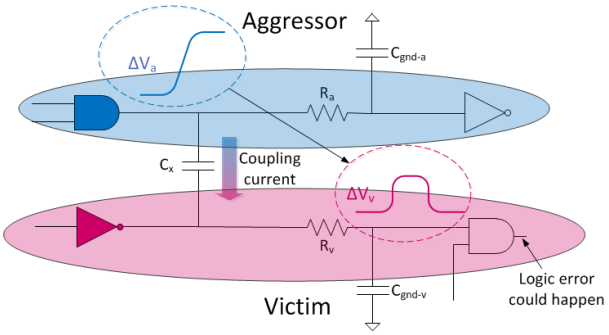


Figure 4.2: Crosstalk noise in a driven victim wire (marked in red) due to coupling current in aggressor-victim capacitance C_x and an aggressor wire (marked in blue). R_a and R_v are the respective resistances, while C_{gnd} values are the capacitances to GND.

with

$$k = \frac{R_a \left(C_{gnd-a} + C_x\right)}{R_v \left(C_{gnd-v} + C_x\right)} \tag{4.3}$$

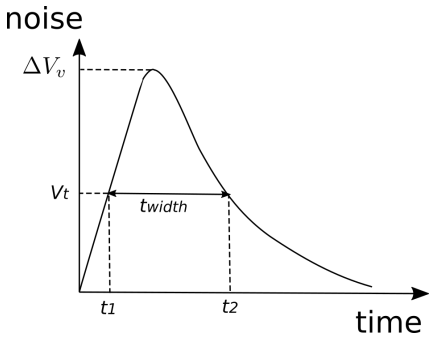


Figure 4.3: Crosstalk peak noise and noise duration on a victim wire. ΔV_v is the maximal crosstalk peak. V_t is the technology dependant threshold voltage used to recognize a logical high. It is exceeded for the time t_{width}.

To clarify the relationship between crosstalk and its mitigations, we utilize the coarse-grained approximation of crosstalk effects shown in Equation 4.2 [155]. Value ΔV_v represents the added voltage induced by crosstalk effects. Values C_x and C_{gnd-v} represent the aggressor-victim capacitance and the capacitance to GND, respectively. Value ΔV_v depends on resistances R_a and R_v and the voltage of the aggressor ΔV_a. The peak noise becomes dependent on the time constant ratio k of the aggressor to the victim [155]. Note that more accurate crosstalk models such as the *L-model* and *pi-model* exist and the interested reader is referred to [80, 42].

Crosstalk Mitigation. Several mitigation strategies have been studied [125, 42] to cope with the undesired effects of crosstalk. For example, the capacitance C can be decreased by (1) shortening the wire length (affecting A in Equation 4.1), or (2) increasing the separation distance (affecting d in Equation 4.1). The delay timing is crucial as multiple aggressors cause an additive effect if transitions occur at the same time. Other crosstalk mitigations include

(3) modifying the driver logic timing with repeaters, (4) increasing the driver for the victim reduces the accumulated crosstalk noise, and (5) alternating adjacent signals between low to high and high to low transitions influences the crosstalk peak ΔV_v and shortens t_{width}.

Unfortunately, from an adversarial point of view, various possibilities remain to induce faults using capacitive crosstalk even if mitigations are applied in the target design.

4.3.2 Design Methodology

We now provide insights on victim wire selection and the use of aggressor routing structures and timing adjustments to deliberately induce a crosstalk effect for reliable exploitation. Figure 4.4 shows the high-level idea of our Trojan design methodology: several aggressor wires are closely routed to the victim wire. The coupling length is the accumulated length in which wires are routed adjacently and closely to produce a practical crosstalk effect.

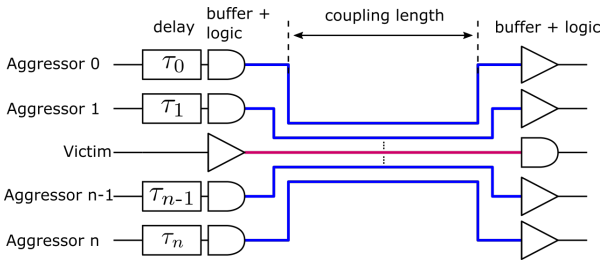


Figure 4.4: Generic model of the proposed hardware Trojan design methodology with several aggressor wires routed close to the victim wire. Based on additive coupling effects, a fault is induced in the victim when all aggressor wires switch at the same time.

Victim Wire Selection

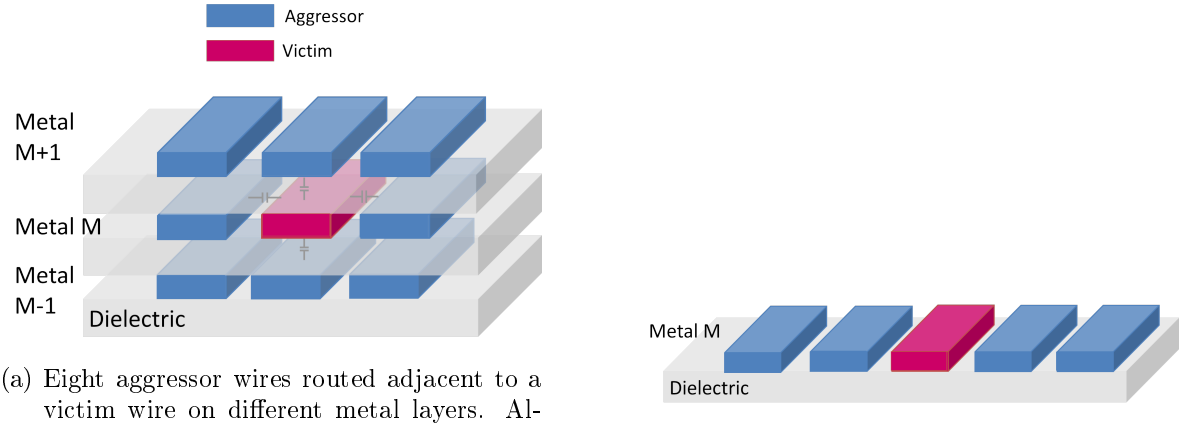
An adversary with the capabilities described in Section 4.2.3 may select almost any number of arbitrary (security-relevant) *victim wires* (e.g., datapath of a cryptographic IP core) for fault injection. The victim wire must be re-routed to achieve a coupling length yielding a crosstalk effect (e.g., 1.25 mm). If the victim wire is in the critical path or its delay is close to the critical path delay, re-routing might cause the wiring delay to exceed the circuit maximum path delay. In such rare cases additional delay corrections to reduce the *parasitic delay* can be applied, e.g. increasing driver strength or re-placing cells to create shorter signal paths.

Aggressor Wires

Several *aggressor wires* must be routed close to a victim wire and switch at the same time to reliably trigger a fault in the victim wire via crosstalk. We now detail several possible routing structures for aggressor wires and strategies for fine-grained delay adjustment. If the aggressor wire is in the critical path or its delay is close to the critical path delay, re-routing may cause an increase in the maximum path delay. The same delay corrections discussed in the previous section can be applied.

Aggressor Wire Routing Structure. We present two strategies with 8 aggressor wires, and 4 aggressor wires routed adjacent to a victim wire, see Figure 4.5. The circuit in Figure 4.5a

does not have alternating (main-)routing directions for a layer nor does it show a correct relationship between a two layer gap and layer specific spacing between two metal lines as this is technology and layer dependent.



(a) Eight aggressor wires routed adjacent to a victim wire on different metal layers. Alternating routing directions between metal layers are omitted for simplicity.

(b) Four aggressor wires routed adjacent to a victim wire on the same metal layer.

Figure 4.5: Aggressor wire routing structure strategies with different numbers of aggressor wires (marked in blue) routed adjacent routed to a victim wire (marked in red) to reliably generate a crosstalk effect in the victim wire.

Our case studies in Section 4.4 and Section 4.5, use the four-aggressor wire routing structure as it implements a reliable Trojan trigger. The layer separation gap is substantially bigger than the intra-layer DRC distance and the alternating routing style of the metal layers[2] prevents long coupling lengths when the common style is not violated. Hence, this favors the use of our 4 aggressor routing wire strategy but leaves a potential for stealthiness improvements with the 8 aggressor strategy, see Section 4.7.

Aggressor Wire Timing Adjustments. Precisely-timed simulations are needed to achieve the simultaneous switching of aggressor wires and thus generate the accumulated crosstalk effect in the victim wire. Since our threat model assumes only the manipulation of the layout but no changes in digital logic gates, only existing wires and their *propagation delays*, also called *flight times*, can be manipulated. We use an RC delay model (Elmore Delay) to describe how to adjust signal arrival times for crosstalk Trojans. Although the RC delay is a simple approximation, it is sufficient for coarse timing approximations and allows for fine-tuning with state-of-the-art simulations. For further information on advanced wire RC delay models and higher order models, we refer the interested reader to [155]. If an aggressor wire is a global input wire, an adversary has to orchestrate a precisely-timed external transition to cause the fault.

To achieve an accumulated crosstalk peak, the *arrival time* of the aggressor charges have to occur at the same point in time. In other words, the difference between the arrival times, called *slack*, has to be close to zero. A positive slack means that the arrival time is too early, while negative slack means that it is too late. Each aggressor wire i is delayed for a specific time τ_i so that $\forall i : \tau_i < \max(\tau_0, \ldots, \tau_n)$ to compensate an additional logic stage in other aggressors or

[2]The alternating routing style switches the primarily routing direction after every layer: Metal-1, Metal-3, ... are routed vertically while Metal-2, Metal-4, ... are routed horizontally to ease the routing.

delay differences within one clock cycle. To realize a zero slack, timing-driven placement and routing algorithms have been proposed [103].

Using a general RC delay model, the non-linear transistor and wire current-voltage (I-V) and capacitance-voltage (C-V) characteristics are estimated with an average resistance and capacitance over a gate's switching range [155]. This model treats a transistor and wire segment as a switch in series with a resistor and abstracts wires with respect to R and C components in different models (e.g., Pi-model, T-Model, ...). Gate and wire segments are *nodes* through which a signal must pass and each element induces a specific delay depending on its I-V and C-V characteristics. An approximation of a required aggressor delay τ_i can be expressed with an Elmore delay model [54] as the sum over each node (gate or wire) j by multiplying the node's capacitance C_i with the resistance R_{ij} on the shared path from the node to the aggressor wire (adjacent to the victim wire):

$$\tau_j = \sum_i R_{ij} C_i \tag{4.4}$$

After defining each τ_i, the adversary selects the (driving) transistor size, wire width, spacing, and layer usage to trade off delay, bandwidth, energy, and noise. He can change the parasitic resistance R and capacitance C of aggressor wires and re-route the layout accordingly. Both resistance and capacitance increase with wire length l, so the RC delay of a wire increases with l^2. Note that the Elmore delay model is only a first approximation in complex modern technologies. Typical Electronic Design Automation (EDA) tools obtain higher accuracy by approximating delays based on higher moments using moment matching techniques (e.g., asymptotic waveform evaluation [155]). An adversary can reuse existing timing analyzer(s) in EDA tools which compute and simulate accurate delay information τ_i for the aggressor wires. Note that so far we have neglected to address process variations. We discuss this topic in Section 4.7.

4.4 Case Study I: Cryptographic Designs

It is necessary that an underlying cryptographic primitive is not manipulated to realize security services such as confidentiality. Since most cryptographic primitives in use are resistant to traditional mathematical attacks, adversaries typically leverage implementation attacks such as SCA and Fault Injection (FI) to disclose key material. Even though SCA and FI countermeasures have been investigated in great detail by the scientific and industrial communities [25], they do not address malicious design manipulations.

In our first case study, we focus on the FI of cryptographic designs by leveraging crosstalk Trojans. Before we detail Trojan design (Section 4.4.1) and implementation (Section 4.4.2), we summarize the general structure of a cryptographic block cipher primitive and the publicly available, third-party AES IP core used in this case study.

General Structure of Cryptographic Designs. Typical cryptographic IP cores employ FSMs to implement the control unit that determines data flows for state and key transformations. An FSM consists of inputs to handle current state and security-critical data path inputs, and control signals to steer the data path. For example, for low area, iterative cryptographic block cipher implementations, security-critical data path signals include round counter information. A manipulated round counter dramatically reduces the security level as multiple round transformations can be skipped eventually resulting in key recovery using (straightforward) cryptanalysis.

Third-Party AES IP Core [65]. In this case study we utilize an open source, third-party AES IP core from OpenCores [65]. This encryption core consists of an iterative round transformation structure and an FSM and a round counter in the control path. As a result, it provides an appropriate case study for cryptographic designs in general.

We deliberately omit gate-level reverse engineering steps for the sake of simplicity. Several works previously targeted datapath reverse engineering for cryptographic designs [140, 61] and control path reverse engineering [126, 99].

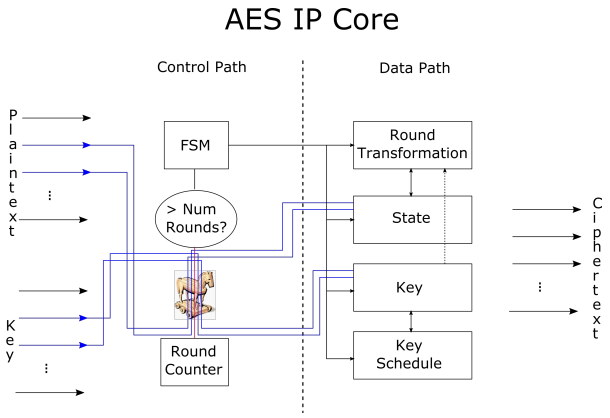


Figure 4.6: Crosstalk Trojan in AES IP core created by rerouting four I/O pad wires (2 plaintext wires and 2 key wires) to cause a crosstalk effect in the control path. On activation, the induced fault causes the last round to toggle yielding state after one round transformation and thus effectively leaking the 128-bit key.

4.4.1 Crosstalk Trojan Design

We now present our crosstalk Trojan for the AES IP core [65].

Victim Wire Selection. We identified a victim wire by manual analysis: `last_round[1]`, see Figure 4.6 to introduce a fault into the round counter and thus leak cryptographic key material. The flag `last_round[1]` indicates the final round of the encryption process, while a counter `v_calculation_cntr[9]` indicates different options and counts the number of bytes processed in each round. In this case study, the victim wire is modified by a fault during computation so that `last_round[1]` is high. The counter `v_calculation_cntr[8]` will independently reach a value above 6 at some point of time which results in the current internal state being sent to the global output for external read-out.

We induce a fault in the first round after the initial AddRoundKey transformation so that the state only consists of the plaintext XORed with 128 bits of the main key. This key can be trivially recovered by XORing the state with the known plaintext. If AES-192 or AES-256 are used, an additional fault occurs after the second AddRoundKey transformation to expose an additional 64 or 128 bits, respectively.

Aggressor Wire Selection. To induce a crosstalk effect into the selected victim wires, we choose the four-aggressor wire routing structure described in Section 4.3.2. Since cryptographic IP cores generally do not contain infrequently-triggered wires, we chose user-controlled global input wires as the aggressor wires. Two plaintext wires (`DATA_I[7]` and `DATA_I[6]`) and two

wires of the `set_key` functionality (`KEY_I[5]` and `KEY_I[4]`) were selected since the IP core expects the key to be present some hundred cycles before the plaintext. The four aggressor wires should never switch simultaneously during normal operation and it is unlikely that the Trojan would be activated by mistake. The accumulated crosstalk noise of each aggressor wire in a narrow time window of several hundred picoseconds (depending on the technology size) results in a voltage higher than the threshold $0.54V$. Following the methodology described in Section 4.3, four aggressor wires and the victim wire were rerouted as $1.25mm$ parallel wires, see Figure 4.7.

Aggressor Wire Timing Adjustments. Our Trojan trigger condition is a timed, specific pattern applied to the four selected I/O aggressor pins. To counter the longer wire delays, the gate driver strength has been adjusted. An encryption process with the targeted (for the adversary unknown) key k_u is performed at the time we trigger the Trojan. AES is started $336ns$ after simulation start. The high-to-low transition of the `VALID_DATA_I` wire in Figure 4.8 shows that the input plaintext is transmitted and the calculation starts. The four aggressor wires are switched at $340ns$ and $345ns$ ($4ns$ and $9ns$ after encryption start) which constitutes an abnormal usage of the IP core. Since the crosstalk Trojan requires transitions to trigger the aggressor wires, they are first set to logic zero before being discharged. When activated, the crosstalk Trojan toggles the *last_ round* D-Flipflop (DFF) to a logic high state at an early encryption stage. This transition leads to a premature ciphertext output that leaks key material. To leak the whole 16-byte key, the pattern must be entered exactly $4.8ns$ after encryption start. The resulting crosstalk noise and Trojan activation are depicted in Figure 4.8.

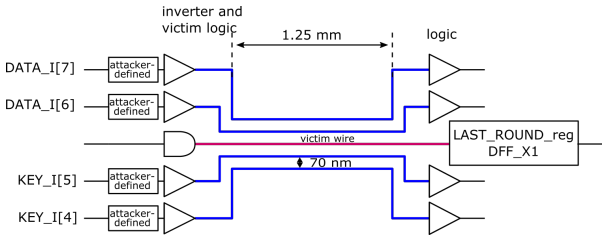


Figure 4.7: Implementation of the crosstalk Trojan in the AES IP core. Aggressor wires (marked in blue) are global inputs, thus the attacker has complete control of the delay elements $\tau_0, \ldots, \tau_3$. These elements can be user-defined after tape-out. The victim wire (marked in red) indicates that the count in the round counter is larger than the number of AES rounds.

4.4.2 Crosstalk Trojan Implementation

We now provide technical details of our crosstalk Trojan.

Place-and-Route. We performed place-and-route using Cadence SoC Encounter version 8.1 to place driver and receiver cells of both aggressor and victim wires $1.25mm$ apart by fixing their gate and wire positions. We deliberately chose this strategy, even though it contradicts the threat model defined in Section 4.2.3, to demonstrate a zero-gate overhead crosstalk-based Trojan since the use of distant driver and receiver cells is likely to appear in typical designs (e.g., bus signals and interfaces).

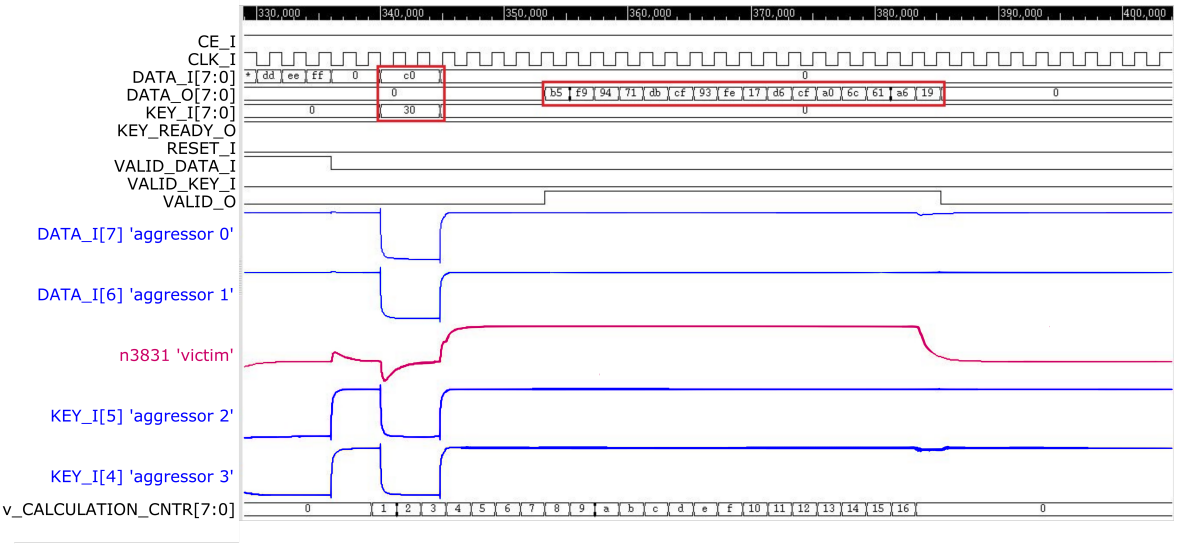


Figure 4.8: SPICE simulation of the triggered crosstalk Trojan for the AES IP core where the victim wire `n3831` exceeds the threshold voltage. We marked the trigger at $t = 340ns$ that discharges the wires, followed by the trigger at $345ns$. The second mark shows the leaked AES state beginning at $353ns$.

Design Verification and Parasitic Extraction. DRC, LVS and Parasitic Exctraction (PEX) checks were verified in Mentor Calibre using rule decks provided by the 45nm FreePDK process design kit [104]. PEX RC parasitics were extracted using hierarchical extraction with generated H-Cells for the standard cells. Since full-chip parasitic extraction is infeasible due to circuit size, we only extracted parasitics for aggressor and victim wires and their respective driver and receiver gates.

Post-Layout SPICE Simulation. We performed post-layout SPICE simulation with the Synopsys HSIMplus Fast SPICE simulator using a low threshold transistor model taken from the FreePDK 45nm predictive technology model. Post-layout simulation of the AES core was performed using the Xilinx ISE design suite. No DRCs or simulation timings were violated.

4.5 Case Study II: OpenRISC 1200

General-purpose CPUs enforce a separation of *kernel mode* code from restricted *user mode* code to mitigate damage caused by vulnerabilities. Since an exploitable bug of a user-mode software application, such as a web-browser, typically results in adversary control of memory, an attacker is then able to execute malicious software on a user's machine. Subsequent *privilege escalation* attacks in which the adversary gains elevated access rights, i.e. kernel model accesses all system resources, are even more damaging.

In our second case study, we focus on FI to realize privileged escalation on a complex, modern processor. Prior to presenting our Trojan design (Section 4.5.1) and implementation (Section 4.5.2) we describe our system model, the adversary's capabilities, and details of our employed

third-party OpenRISC 1200 processor IP core. This case study was inspired by Yang *et al.* [158] who realized an analog Trojan to elevate privilege rights for a OpenRISC 1200 processor.

System Model and Attacker Capabilities. We assume the following typical system model for the OpenRISC 1200 processor: the adversary is able to leverage an exploitable bug in the software running on the OpenRISC 1200 to gain an arbitrary memory write primitive (e.g., by a classical buffer overflow). The adversary writes a specific user-mode instruction triggering an a priori inserted crosstalk Trojan to elevate privilege rights.

Third-Party OpenRISC 1200 Processor [107]. The OR1200 is an open source IP core. The publicly-available design is written in Verilog HDL. The implementation includes a power management unit, debug unit, tick timer, programmable interrupt controller, central processing unit (CPU), and memory management hardware. Peripheral systems and a memory subsystem may be added using a standardized 32-bit Wishbone bus interface. The OR1200 has performance comparable to an ARM10 processor architecture. Generally, the OR1200 is intended for use in a variety of embedded applications, including telecommunications, portable media, home entertainment, and automotive applications. A GCC toolchain and a Linux kernel port for OR1K, which runs on the OR1200, have been implemented.

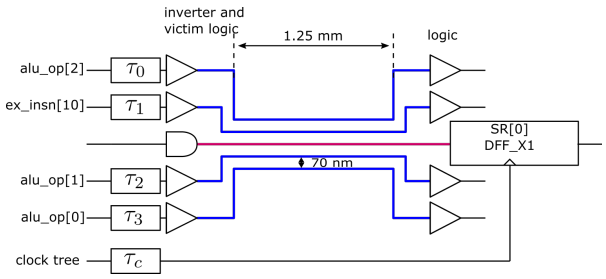


Figure 4.9: Implementation of the crosstalk Trojan in an OpenRISC 1200 IP core. Aggressor wires (marked in blue) are controlled by an adversary-crafted instruction, respective delay elements $\tau_0, \ldots, \tau_3$ did not require any adjustments. We adjusted delay element τ_c to $0.2ns$ for the clock tree of the SR[0] FF indicating user mode (= 0) or supervisor mode (= 1). See Section 4.5.2 for further details on clock tree adjustments.

4.5.1 Crosstalk Trojan Design

We now present our crosstalk Trojan for the OR1200 CPU.

Victim Wire Selection. Since the processor mode (e.g., user or supervisor mode) indicator of the OpenRISC 1200 is stored in a bit of the *supervisor register* [107], i.e. `SR[0]`, it is targeted to force a privilege rights elevation that is similar to Yang *et al.* [158]. This flag grants supervisor rights (arbitrary use of the CPU) when set. Thus, once a user mode application can be exploited, a fault is induced in the supervisor register to obtain supervisor rights.

Aggressor Wire Selection. The four-aggressor wire routing structure described in Section 4.3.2 was chosen to induce a crosstalk effect in the selected victim wire. Since general-purpose CPUs include various rarely-switched wires related to undefined and reserved instruction fields, the `ff1` (find first) instruction was selected as it can be executed with user mode rights and consists of unused instruction fields, see Figure 4.10. Three opcode wires (`alu_op[0]`, `alu_op[1]`, `alu_op[2]`) were used to identify the instruction. A reserved instruction bit field

(`ex_insn[10]`) from the `ff1` instruction was also used in the attack. Since typical assemblers set the reserved bit field to 0, the Trojan is generally not activated by mistake. As a result, the execution of a genuine `ff1` instruction does not trigger the Trojan. Following the methodology described in Section 4.3, we re-routed the four aggressor wires and the victim wire as $1.25mm$ parallel wires, see Figure 4.9.

l.ff1	Find First 1						
opcode 0x38	D	A	reserved	reserved	opcode 0x0	reserved	opcode 0xf
6 bits	5 bits	5 bits	5 bits	1 bit	2 bits	4 bits	4 bits

Figure 4.10: OpenRISC 1200 `ff1` instruction [107]. The reserved Trojan trigger bit is marked in red.

Aggressor Wire Timing Adjustments In contrast to the previous case study, we are not able to directly change the timing of the internal aggressor wires. However, our chosen aggressor wires are all in the same vicinity and logical stage and thus require no further timing adjustments.

Victim Wire Timing Adjustments. Since the destination DFF of the victim wire has an early rising-edge, it might miss the noise peak of the crosstalk Trojan. Hence, it is necessary to manipulate the internal delays, especially the setup and hold times of the victim DFF. To create a fault, setup and hold times are delayed to coincide with the crosstalk noise peak. The shift can be seen in Figure 4.11. A clock delay can be introduced by changing the clock tree (e.g., rerouting towards the DFF). To set the required delay, we leverage the propagation delay of existing clock buffers and increase/decrease the skew with a delay macro in Cadence SoC Encounter. In our case, we shifted the DFF clock by $0.2ns$ and re-routed the design. Note that the SoC Encounter macro might add further gates (e.g., buffers) or change existing gate sizes for timing adjustments. The resulting crosstalk noise and Trojan activation are depicted in Figure 4.12.

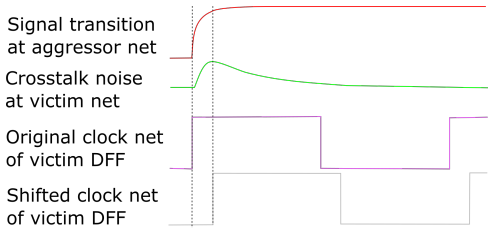


Figure 4.11: The shift in clock net timing for the victim DFF.

4.5.2 Crosstalk Trojan Implementation

We now provide technical details of our crosstalk Trojan.

Place-and-Route. Similar to the approach used in Section 4.4, place-and-route was performed using Cadence SoC Encounter version 8.1. We inserted the Trojan after tape-out, reran

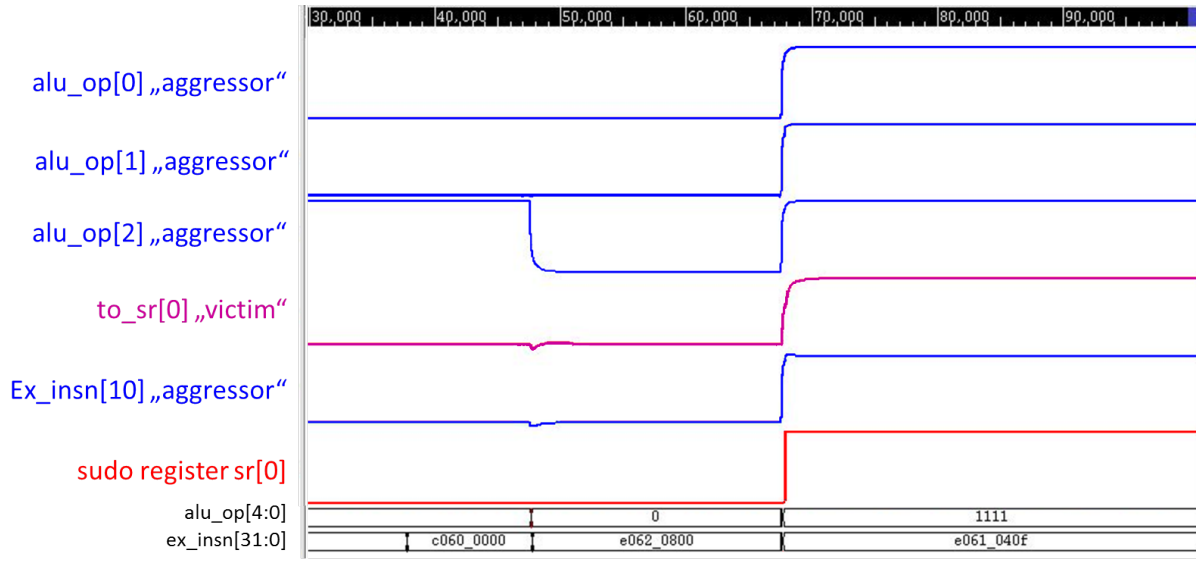


Figure 4.12: SPICE simulation of the triggered crosstalk Trojan for the OR1200 IP core. The victim wire `to_sr[0]` exceeds the threshold voltage and switches the supervisor mode register (SR[0]) at $t = 67ns$.

place-and-route using the back annotated netlist and ensured that original timing constraints were met. The design was synthesized using the Nandgate $45nm$ open cell library with a clock frequency of $200MHz$ and Wishbone interfaces for data and instructions. External debug port and power management interfaces were included.

Design Verification and Parasitic Extraction. DRC, LVS and PEX were verified in Calibre using rule decks provided by the $45nm$ FreePDK process design kit. PEX RC parasitics were extracted using hierarchical extraction with generated H-Cells for the standard cells. Since full-chip parasitic extraction was infeasible due to circuit size, we only extracted parasitics for aggressor and victim wires and their respective driver and receiver gates.

Post-Layout SPICE Simulation. We simulated the OpenRISC 1200 using the Wishbone bus interface with Synopsys HSIMplus Fast SPICE simulation based on the freePDK 45nm predictive technology model. We verified that switching three out of four aggressor wires does not yield a sufficient accumulated crosstalk peak to accidentally trigger the Trojan. A fault in the victim wire is only induced by switching all four wires.

4.6 Mitigating the Risk of Parasitic Trojans

The class of parametric Trojans, including our crosstalk Trojan, leverages parasitic effects with negligible, close to zero gate overhead. These Trojans hide within place-and-route layout information. As we have shown in Section 4.2.2, current IC reverse engineering strategies abstract parasitic and geometrical wire information and thus cannot identify parasitic hardware Trojans automatically.

Our proof-of-concept crosstalk Trojans are built from long straight wires. Finding such long wires by manual visual inspection is possible, although it is a daunting and time-consuming

task for a million-node GDS layout. Moreover, the problem will be more challenging for future reduced technology sizes, see Section 4.6.2.

We now provide insights into how crosstalk Trojans may be detected in practice using improved reverse engineering strategies. In general, this problem is addressed by retaining parasitic information after a vectorized design is obtained, see Figure 4.13.

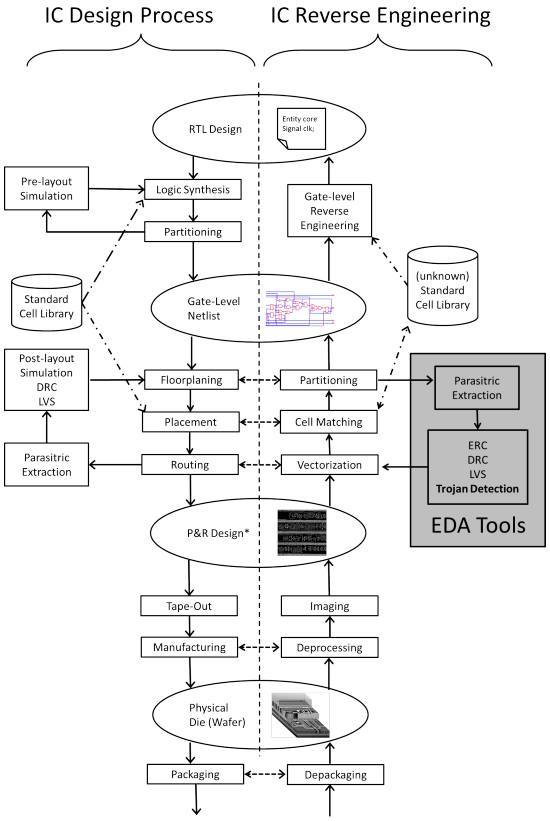


Figure 4.13: Improved IC reverse engineering with additional steps to analyze parasitic information (marked in gray). Parasitic extraction and Trojan detection can be realized with existing EDA tools.

Parasitic Extraction. The major purpose of parasitic extraction in IC design is to generate an accurate (analog) circuit model circuit so that digital and analog circuit responses can be simulated. Digital circuit and cell responses are analyzed and simulated for delay and loading calculations such as timing analysis, power analysis, circuit simulation, and integrity analysis. Analog circuits are often run using detailed test benches to indicate if extra extracted parasitic information allows a designed circuit to continue to function normally. Parasitic extraction is typically infeasible for modern layout sizes (depending on design size and simulation details). Although *field solvers* provide physically accurate solutions, they are only applicable to small designs or design parts due to high computational effort. For example, annotated parasitic models can be simulated with SPICE on the scale of several thousands of nets. Hence, approximated pattern matching solutions are commonly used in commercial tools for modern integrated circuit designs. Nevertheless, they become inaccurate for smaller technology sizes such as $32nm$ and

$28nm$ [34]. In the scope of a multi-million net IC design, a previous separation or partitioning is mandatory to simulate *suspicious* areas for accurate parasitic Trojans. This approach implies that parasitic extraction can only support reverse engineering by finding *known* parametric Trojans if we simulate a reduced ROI. In the following section, we provide a proof-of-concept detection technique for the crosstalk Trojan.

4.6.1 Parametric Crosstalk Trojan Detection

Detection Model Capabilities. We assume that the reverse engineer has access to the unannotated layout to detect crosstalk Trojans. Such access capability occurs for (1) chip-level reverse engineering (see Section 4.2.2), and (2) access to the layout by an untrusted IP provider. To estimate parasitic crosstalk capacitances we require HSPICE models, however, accurate HSPICE models are typically not available as they are fabrication trade secrets. Moreover, HSPICE simulations are infeasible for modern IC designs due to CPU intensive calculations. As a result, we must employ heuristics to identify suspicious areas where crosstalk Trojans may be implemented.

Suspicious Wire Extraction. Since manual visual inspection of modern IC designs with a technology size of $45nm$ or smaller is practically infeasible (due to the vast amount of wires), we aim to identify the suspicious wires of a crosstalk Trojan in an automatic fashion. To this end, the reverse engineer analyzes the layout and respective SPICE model by ranking all wires by length. Long wires are characteristic for parametric Trojans based on crosstalk, as described in Figure 4.1 and Equation 4.1. In practice, this ranking step can be performed within hours for typical IC designs.

Interconnection Capacitances Estimation. After wire ranking, the reverse engineer performs a parasitic extraction and identifies whether any capacitance value exceeds the average case by a certain threshold, i.e. 200%. In this process, the reverse engineer determines the interconnection capacitance matrix between individual entries on the n longest wires to identify characteristic capacitances of crosstalk Trojans. As the capacitance depends on the technology used, thresholds on the average value of long wires are approximated rather than being set to fixed numbers. Millions of nets can be analyzed with this capability once the geometric structure is exported [56].

Proof-of-Concept Detection. To identify the crosstalk Trojan described in Section 4.5, the GDSII layout was directly employed and the *deprocessing*, *stitching*, and *image processing* steps were omitted for the sake of simplicity. Even though multiple commercial third-party tools for layout edit and extraction are available, not every reverse engineer has access to these tools and, thus the freely available *KLayout* tool was used to extract geometries and materials from the layout, and *FasterCap* was used to estimate the capacitances.

First, wires were ranked according to their length and the $n = 100$ longest wires were extracted. This number was limited to 100 due to high computation effort requiring several hours on a standard laptop. Aggressor wires were ranked the 6^{th} to 10^{th} longest. Only clock sub-trees and several data buses were longer. Second, the interconnection capacitance matrix for the $n = 100$ longest entries was determined, and each wire where the capacitance value is higher than 200% above the average was marked as suspicious. All 5 wires which are part of the crosstalk Trojan (4 aggressor wires and 1 victim wire) were identified. Afterward, we simulated the 5 isolated wires with inaccurate SPICE models and identified the crosstalk fault manually. The faulted signal is the *sr[0]* register signal, which leverages the user's rights.

In summary, we were able to identify crosstalk Trojans in an automated manner. We neither optimized coupling lengths nor obfuscated interconnects, as described in Section 4.7 . Once coupling lengths are shortened by around 20%, coupling wires become challenging to identify by only analyzing their length.

4.6.2 On Sophisticated Parasitic Trojans

Similar to other hardware Trojan obfuscation schemes [160], several strategies can be used to increase the stealthiness of characteristic long aggressor wires. Obfuscation strategies which hamper our proposed mitigation are now described.

Increased Routing Complexity. Aggressor and victim wires can be routed with complex structures (corners or curves), through multiple layers, or split-up and re-emerge at multiple points. An adversary can approximate the coupling capacitance to the victim with sophisticated tools. This capacitance can be kept constant during complex routing. Furthermore, dense adjacent wires can be camouflaged as a bus by increasing/decreasing the numbers of aggressor wires or inserting dummy wires to obtain a standard bus width (e.g., 8, 16, and 32 wires).

Victim Threshold Decrease. Another way to decrease the required coupling length is to change the threshold voltage of the victim gate by doping or body biasing. Dual-Vt design is common in ICs and allows transistors to be designated as high or low threshold devices. Low threshold devices are fast and used where the delay is critical, and high threshold devices are slow and used elsewhere to reduce static power [66].

Increased Aggressor Voltage. Capacitive coupling noise is increased with an increased aggressor voltage $\Delta V_{aggressor}$, as seen in Equation 4.2. An adversary might pick a power supply or charge pump, often required for analog circuits, or memory cells, as aggressors. For example, EEPROM and flash circuits require high voltage to erase memory cell states.

DRC Violations. In special scenarios, an adversary might be able to violate DRC without raising suspicions. In-house fabrication or malicious insiders decrease the wire separation gap. This is an effective way to increase capacitive crosstalk, as discussed in Section 4.3.1.

Dielectric Changes. Changing the dielectric between the routed aggressors and the victim has an immediate effect on the ratio of C_x and the respective ground capacitance. In a visual inspection process, a reverse engineer needs at least one additional step to see local changes in the dielectric. With the additional hardware reverse engineering step, the reverse engineer has to trade off computation power versus extracting the longest wires and simulating their capacitance matrix, if the wire length can be further shortened. Hence, the arms race between hiding and finding (crosstalk) Trojans continues.

4.7 Discussion

Generality and Impact. We have demonstrated that a capacitive crosstalk Trojan can be used as a targeted fault injection primitive in two offensive use cases for a modern $45nm$ technology size. We have also demonstrated the leakage of sensitive cryptographic key material and mounted a privilege escalation attack on a modern CPU. Since only the layout of the chip was changed, the standard IC design flow was not interrupted or altered. To this end, the adversary must know the position and meaning of the faulted signal (e.g., by reverse engineering).

An adversary can circumvent fault injection mitigation techniques in the data and control paths with additional (crosstalk) Trojans as he has white-box access to the layout. This step may require additional reverse engineering effort.

General Adversary Proceeding. As stated in our threat model, cf. Section 4.2.3, we assume the possibility of (re)placing-and-routing the layout and approximating the technology files. In principle, there are two distinct attack points: (1) before tape-out by a malicious designer, and (2) after tape-out by a malicious foundry.

In the former case, a malicious designer may insert a crosstalk Trojan during the initial place-and-route, whereas in the latter case, a malicious foundry is faced with the challenge of (re-)generating a malicious layout while maintaining the original functionality. In particular, an adversary may keep original placement information and only re-route the design after Trojan insertion. As a result of an additional re-routing step, existing cells might be changed (e.g., due to gate re-sizing) or gates might be added (e.g., buffers to meet timing constraints). These changes do not affect the logic function of the abstracted gate-level netlist as discussed in Section 4.6.

To demonstrate the feasibility of zero-gate overhead crosstalk-based Trojans, we utilized a pre-placed layout in the AES case study (Section 4.4) and added manual straight routes for the Trojan. The OR1200 case study (Section 4.5) shows the practicality of crosstalk-based Trojan insertion into a fully placed-and-routed large-scale hardware design.

Stealthiness. To main consistency with existing literature, we define the term *stealthiness* as a property that reflects stealth during testing and operation. Since crosstalk Trojans are triggered by adversary-selected aggressor wires, wire selection directly affects Trojan stealthiness. To maximize the stealthiness, we must minimize the probability that all aggressor wires switch simultaneously by accident. For example, in modern complex CPU designs, an adversary can access a variety of internal wires in exception handling units, power management units, or debugging units that could serve as potential aggressor wires. Furthermore, primary inputs may be selected as aggressors that must be activated in a narrow time frame to create an additive effect. In the presence of pipelining (e.g., for an unrolled cryptographic implementation or processor pipelines), it is possible to stretch an instruction trigger sequence over multiple clock cycles. A technique by Ghandali *et al.* [66] can be used to automatically identify rarely switching paths, i.e. potential aggressor wires.

Reliability. Generally, the adversary tries to maximize both stealthiness and reliability. In our use cases, we utilized 4 aggressor wires to increase the stealthiness while we assume the process variations (5% capacitance changes) to be insignificant for the reliability. For the unlikely case that process variations become a practical hurdle, the adversary can increase the crosstalk noise with the discussed design methods (e.g., increase the wire length, see Section 4.3) until the effect is reliable again. Note that for our parameters a higher number of aggressor wires will not have a significant beneficial impact on the induced crosstalk peak as the distance to the victim wire is the major limiting factor. When aggressor 1 to 4 are switching simultaneously, they generate a noise peak as close to V_t while the 5^{th} aggressor wire contribution is supposed to push the peak above. Since the 5^{th} contribution is small, it may become unreliable with worst-case process variations and environmental changes. Hence, adding a 5^{th} aggressor wire decreases the reliability, limiting us to 4 aggressor wires to balance stealthiness and reliability for our parameters.

Generally, an adversary tries to maximize both stealthiness and reliability. In our use cases, we utilized four aggressor wires to increase the stealthiness while assuming that process variations

(10% capacitance changes) are insignificant for reliability. In the unlikely case that process variations become a practical hurdle, the adversary can increase the crosstalk noise with the discussed design methods (e.g., increase the wire length, see Section 4.3) until the effect is reliable again. For our parameters, an increased number of aggressor wires will not have a significant beneficial impact on the induced crosstalk peak as the distance to the victim wire is a major limiting factor. When aggressors 1 to 4 are switching simultaneously, they generate a noise peak close to V_t while a 5^{th} aggressor wire contribution would not push the peak much higher. Since the 5^{th} contribution is small, it may become unreliable with worst-case process variations and environmental changes. Hence, adding a 5^{th} aggressor wire decreases the reliability, limiting us to four aggressor wires to balance stealthiness and reliability.

4.8 Conclusion

In this paper, we leveraged the parasitic effects of closely routed adjacent interconnects to design parametric malicious circuits, a crosstalk Trojan. Since these Trojans can be realized by only rerouting existing interconnections, they can possess a zero gate overhead which is both stealthy and hard to identify with standard visual inspection techniques. In two case studies, we demonstrated that crosstalk Trojans provide reliable, adversary controllable faults to (1) insert malicious circuitry in a cryptographic AES IP to leak the secret key and (2) elevate user access rights to allow arbitrary code execution in ring 0 for a modern and fully-fledged CPU. We then investigated how the currently employed visual inspection hardware reverse engineering workflow has to be enhanced to cope with parametric hardware Trojans in a semi-automatic fashion. To this end, we developed a novel automated layout-level mitigation which exposes characteristic wire lengths of crosstalk Trojans. Finally, we highlighted potential implementation strategies that must be considered for future hardware Trojan detection.

Since we believe that crosstalk Trojans are a building block of stealthy and hard-to-detect hardware Trojans, we recommend that the research community includes parametric characterizations in the hardware reverse engineering process to detect such advanced hardware Trojans.

Part III

Real-World SoC Embedded Security Analysis

Chapter 5

Microcode Mask ROM Extraction from a modern CPU

Modern x86 processor microcode is an abstraction layer that interprets user-visible CISC instructions to hardware-internal RISC microinstructions. The microcode architecture is the closest partially programmable engine to the hardware, with unknown security control mechanisms. The purpose is to enable the vendors to modify CPU behavior in-field, and thus patch erroneous microarchitectural hardware processes or even implement new features. Unfortunately, microcode is proprietary and closed-source. It is closely hardware-coupled and hard-wired on the die creating a big hurdle for the security community without access to hardware reverse engineering equipment. In this chapter we overcome this hardware hurdle and extract a complete microcode ROM. We explain the structure and reconstruct its physical mapping to understand the ROM content. This reveals insights into the proprietary CPU innings and shows a sophisticated, hybrid ROM implementation. This chapter is based on the two publications [88, 87] in a joint work with Koppe, Kollenda, Fyrbiak and Gawlik.

Contents of this Chapter

Contribution: *In the context of this project, my contribution is the deprocessing of the K8 AMD CPU and the analysis of the microcode Read-Only Memory (ROM) to retrieve valid microcode instructions with their initial interleaved sequence from a hardware point of view. Finally a physical mapping from the corrected virtual address space to hardware ROM has been done.*

5.1 Introduction

Similar to complex software systems, bugs exist in virtually any commercial CPU hardware and can cause erroneous behaviour [41, 157] that leads to even severe consequences on system security, e.g., privilege escalation [50, 96] or leakage of cryptographic keys [27, 86, 95, 73, 91, 31]. As hardware bugs are hard to patch after tape-out, errata sheets from embedded to general-purpose processors inform and warn about incorrect behavior with accompanying workarounds to prevent hardware damage and safeguard program execution [75, 7]. Such notes contain instructions for developers on how these bugs can be bypassed, mitigated or improved e.g., by means of recompilation [101] or binary re-translation [36]. However, these interim solutions are not always suited for complex hardware design errors which require hardware modifications [118]. Dedicated hardware units to counter bugs are imperfect [96, 119] and involve non-negligible hardware costs [17]. Please note, that hardware bugs can be encountered in any electronic system containing IC hardware.

Two publicly known hardware bugs, throughout the years, are the infamous *Pentium fdiv* bug [157] and the recently shown Spectre and Meltdown [86, 95, 73, 91, 31]. They highlight the importance of microcode updates in a complex CPU. Hardware bugs in CPUs are usually found by debugging erroneous behavior[157] or detected during academic research[86, 95, 73, 91, 31]. After deployment, any hardware is fixed and necessary patches become expensive with IC device numbers in the millions. It shows a clear economic need for field updates in order to turn off defective parts and patch erroneous behavior, especially when system security or humans safety is in danger. Note that the implementation of a modern processor involves millions of lines of HDL code [139] and verification of functional correctness for such processors is still an unsolved problem [75, 7].

Since the 1970s, x86 processor manufacturers have used microcode to decode complex instructions into series of simplified microinstructions for reasons of efficiency, optimizations, and diagnostics [112]. From a high-level perspective, microcode is an interpreter between the user-visible Complex Instruction Set Computer (CISC) Instruction Set Architecture (ISA) and high-performance internal hardware based on Reduced Instruction Set Computer (RISC) paradigms [132]. Initially, microcode was implemented in read-only memory, although manufacturers introduced an update mechanism by adding a microcode Random Access Memory (RAM) for patches.

Once an erroneous CPU behavior is discovered e.g. the recent Spectre and Meltdown, manufacturers publish a microcode update, which is loaded through the Basic Input/Output System (BIOS)/Unified Extensible Firmware Interface (UEFI) or OS during the boot process. Due to the volatile nature of the patch RAM, microcode updates are not persistent and have to be reloaded after each processor reset or boot. With the possibility of microcode updates, processor manufacturers obtain the flexibility and reduce costs of correcting erroneous behavior with few additional hardware resources. Note that both Intel and AMD deploy a microcode update mechanism since Pentium Pro (P6) in 1995 [37, 76] and K7 in 1999 [37, 5], respectively. Unfortunately, CPU vendors keep information about microcode secret. Publicly available documentation and patents merely state vague claims about how real-world microcode *might* actually look like, but provide little other insight. A throughout bug and security evaluation is not possible due to the big hurdle of hardware reverse engineering. Necessary expensive equipment and delayering experience limit the researchers access to the internal microcode implementation. Research in

hardware reverse engineering has mostly been done on PCB scale systems or small microcontrollers. Nevertheless, hardware faults and bad (crypto-)designs have been found [133, 81] in the past.

Goals. In this chapter, we focus on the hardware ROM of the microcode in a modern x86 CPUs and our goal is to answer the following research questions:

(1) What is microcode and what is its role in x86 CPUs?

(2) How does the microcode update mechanism work?

(3) How is the microcode hardware (ROM) structured?

(4) What is the content of the microcode ROM?

In order to answer Question (1) and (2), we emphasize that information regarding microcode is scattered among many sources (often only in patents). Hence, an important part of our work is dedicated to summarize this prerequisite knowledge. Furthermore, we tackle shortcomings of prior attempted security analyses of x86 microcode, which were not able to reverse engineer microcode [37, 11], neither from the software nor the hardware side. After we obtain a detailed understanding of the x86 microcode for several CPU architectures and its performance, we can derivate hardware assumptions to locate and reverse engineer the microcode implementation. This allows us to tackle the hardware hurdle and answer questions (3) and (4). As a result, we obtain an understanding of the inner hardware of CPU updates that would, in the end, allow us to even generate our own updates that focus on potential applications of microprograms for both defensive and offensive purposes.

Our analysis focuses on the AMD K8/K10 microarchitecture since these CPUs do not use cryptographic signatures to verify the integrity and authenticity of microcode updates. Note, that Intel started to cryptographically sign microcode updates in 1995 [37] and AMD started to deploy strong cryptographic protection in 2011 [37]. We assume that the underlying microcode update mechanism is similar, but cannot analyze the microcode updates since we cannot decrypt them.

Contributions. In summary, our main contributions in this chapter are as follows:

- **In-depth Analysis of Microcode.** We provide an in-depth overview of the opaque role of microcode in modern CPUs. In particular, we partly reverse engineer the fundamental ISA of a microcode including the updates as deployed by vendors to patch CPU defects and errors.

- **Hardware Reverse Engineering.** We show the hardware reverse engineering findings of a highly optimized hardwired mask ROM in a modern, fully-fledged CPU. The extensive optimizations are partly known from modern Dynamic random-access memory (DRAM) structures, transferred to masked ROM. We successfully retrieved single microinstructions from the masked ROM.

- **Physical Mapping.** We face the problem of finding sequential microcode instructions by emulation and show the result of the mapping directly on the underlying physical mask ROM. This enables us to read the microcode operation implementation.

5.2 Background and Related Work

In the following, we first provide the background information needed to understand the microcode and gain basic knowledge for the underlying hardware presented in this chapter. In addition, we discuss related work that demonstrated mask ROM readout and the capabilities of microcode.

5.2.1 Microcode Background

Microcode is part of the abstraction layer between the outward facing, complex instruction set and the inner, simpler hardware-bound architecture of CPUs [132]. During the decoding of an instruction, different decoders can be used: ① a direct (or short) path decoder, handling simple and fast instructions, ② a long decoder, handling more complex instructions and ③ the vector (or microcode) decoder, handling the most complex and seldomly used instructions. The direct and long decoder are directly implemented in hardware, while the vector decoder uses a software-assisted design. For the rest of this chapter, we call instructions implemented by the direct or long decoder *direct path instructions* and those implemented with the help of microcode *vector path instructions*. The reason for the hybrid approach is that hardware decoders are inflexible and costly in hardware, especially in regard to die size, but offer a faster translation compared to the microcode decoder, which is clocked and far more complex. Another feature that is realized with the help of microcode is the ability to update the microcode and thus patch erratas in CPUs without the need to resort to expensive product recalls and fabrication mask updates [98, 88]. This update functionality is used by Intel, AMD and ARM processors. The vendors provide the microcode updates and details on how to apply them to BIOS or UEFI vendors and OS developers. The updates are loaded shortly after the start of the system using a dedicated Model-specific register (MSR). Updating the microcode requires kernel privileges. On modern CPUs the updates are (believed to be) protected by strong public key cryptography.

5.2.2 Related Work

Before presenting the results of our reverse engineering process, we briefly review existing literature on microprogramming, ROM extraction and related topics.

Microprogramming. Since Wilkes' seminal work in 1951 [156], numerous works in academia as well as industry adopted and advanced microprogrammed CPU designs. Diverse branches of research related to microprogramming include higher-level microcode languages, microcode compilers and tools, and microcode verification [8, 112, 141]. Other major research areas focus on optimization of microcode, i.e., minimizing execution time and memory space [79]. In addition, several applications of microprogramming were developed [69] such as diagnostics [102].

Since microcode of today's x86 CPUs has not been publicly documented yet, several works attempted a high-level security analysis for CPUs from both Intel and AMD [37, 11]. Even though these works reported the workings of the microcode update mechanism, the purpose of fields within the microcode update header, and the presence of other metadata, none of the works was able to reverse engineer the essential microcode encoding. Hence, they were not able to build microcode updates on their own. Nevertheless we emphasize that [11] has a good understanding of the microcode hardware, as even a helpful die shot from the K8 is included.

We note that Arrigo Triulzi presented at TROOPERS'15 and '16 that he had been able to patch the microcode of an AMD K8 microarchitecture [146, 147]. However, he did neither publish the details of his reverse engineering nor the microcode encoding.

Imperfect CPU Design. Although microcode updates can be leveraged to rectify some erroneous behavior, it is not a panacea. Degrading performance with microcode updates due to additional condition checks are willingly accepted even though they cannot be applied in all cases. An infamous example is AMD's K7, where the microcode update mechanism itself was defective [37, 5]. In order to tackle these shortcomings, diverse techniques have been proposed including dynamic instruction stream editing [43], field-programmable hardware [119], and hardware checks [17, 96].

Trusted Hardware. The security of applications and operating systems builds on top of the security of the underlying hardware. Typically software is not designed to be executed on untrusted or potentially malicious hardware [48, 27, 50]. Once hardware behaves erroneously (regardless of whether deliberately or not), software security mechanisms can be invalidated. Numerous secure processors and Trusted Platform Modules (TPMs) have been proposed over the years [138, 97, 45]. Commercially available examples include technologies such as Intel SGX [44], AMD Pacifica [6] and ARM Trustzone [14].

However, the periodicity of security-critical faults [75, 7] and undocumented debug features [50] in closed-source CPU architectures challenges their trustworthiness [44, 117].

5.2.3 Mask ROM Readout

Mask ROMs are used in high volume uCs and CPUs that sacrifice the flexibility of the memory content for a better memory density. In large volume production, this cuts costs and results in a better die yield. Dedicated stand-alone mask ROMs chips can be bought and are mostly found in old videogame cartridges. These chips can be read electrically and do not offer any kind of security. Note however that because such chips are easy to read they often are supplemented by security processors to prevent trivial cloning. Further advanced mask ROMs can be encrypted or hard to reach due to modern IC manufacturing processes with up to 13 layers. Hardware reverse engineering ROM is, therefore, depending on the processes and materials used in manufacturing as shown in Chapter 2. The ROM readout is hard to automate, resulting in projects distributing ROM pictures to multiple users for manual readout[1]. Nevertheless, (half-)automated image processing solutions[2, 13] exists to reduce the process overhead.

Prior (academic) ROM read-out has been done optically on small uCs or dedicated ROM memory. This reduces preparation effort as the masked ROM is somewhere in the top layers. Small dies from uCs have few layers that can be removed with a single wet chemical deprocessing step[12, 13, 2, 131] and have a single ROM block, built in a single memory type. Note that some memory types require additional preparation steps e.g. staining[131].

Prior to delayering the microcode memory structure of the AMD K8 in Section 5.4, we expected a ROM structure, as a highly optimized, fast memory structure that is build as small as possible without any performance drawbacks. This saves space on the die during manufacturing resulting in a smaller die and a better yield. This kind of memory can only be read and never altered. Thus, we expected *NAND* or *NOR* mask ROM.

In this chapter, the DUT is a modern, fully-fledged CPU with around 10 layers and sophisticated technological optimizations with unknown content. The technology size is 130nm.

Please note that this technology size is hard to resolute with optical microscopes and requires a precise, accurate delayering process. With regard to the ROM type that has been identified in the DUT, we refer the interested reader to the *NOR* mask ROM by contact layer in the appendix C . A comprehensive collection of ROM types can be found in [131]. The extraction of the *NOR* ROM contact layer can be done optically if the deprocessing to reach the respective layer is adequate.

5.3 Reverse Engineering Microcode

In this section, we provide an overview of the AMD K8 and K10 microarchitecture families and revealed by our reverse engineering approach. Furthermore, we present our analysis setup and framework that includes prototype implementations of our concepts and supported our reverse engineering effort in a semi-automated way. This section is not the main contribution of this thesis. Hence, the software approach is briefly described to give basic insight necessary for the hardware reverse engineering.

Our software sided analysis primarily covers AMD K8 and K10 processors because—to the best of our knowledge—they are the only commercially available, modern x86 microarchitectures lacking strong cryptographic protection of microcode patches.[1]

5.3.1 AMD K8 and K10

AMD released new versions of its K8 and K10 processors from 2003 to 2008 and 2008 to 2013, respectively. Note that the actual production dates may vary and in 2013 only two low-end CPU models with K10 architecture were released. K9 is the K8's dual-core successor, hence the difference is marginal from our point of view. Family 11h and 12h are adapted K10 microarchitectures for mobile platforms and APUs.

All of these microarchitectures include a microcoded Instruction Decode Unit (IDU). The x86 instruction set is subdivided into *direct path* and *vector path* macroinstructions. The former mainly represent the frequently used, performance critical macroinstructions (e.g., arithmetic and logical operations) that are decoded by hardware into up to three microinstructions. The latter are uncommon or complex, and require decoding by the microcode sequencer and are stored in microcode ROM. Vector path macroinstructions can produce many microinstructions and even loops are possible. During execution of the microcode sequencer, hardware decoding is paused. The microcode is structured in *triads* of three 64-bit microinstructions and one 32-bit sequence word [37].In 2002 AMD described an example microinstruction in patent RISC86 [57]. The sequence word indicates that decoding is complete or contains the address of the next triad. The microcode ROM is addressed in triad-length steps. Thus, an example address space ranging from `0x0` to `0xbff` contains 3,072 triads. The microcode, stored in the ROM, is responsible for the decoding of vector path macroinstructions and handling of exceptions, such as page faults and divide-by-zero errors.

[1] In the context of this section, my contribution the analysis of the microcode ROM to retrieve valid microcode instructions with their initial interleaved sequence from a hardware point of view. Additionally a physical mapping from the corrected virtual address space to hardware ROM has been done.

5.3.2 Update Mechanism

The K7, released in 1999, is AMD's first microarchitecture supporting microcode updates. AMD kept the update feature secret until it was exposed along with three K8 microcode patches in 2004. The update mechanism did not change throughout to the 12h family. The patches and the update mechanism were reverse engineered from BIOS updates [11]. Microcode updates are stored in a proprietary file format, although pieces of information have been reverse engineered [37, 11]. Starting with the K10 microarchitecture, AMD started to publicly release microcode updates, which benefits the Linux open-source microcode update driver. Our view of the file format is depicted in Table 5.1 including the header with checksum and number of triads, match register fields, and triads. It should be noted that triads in microcode updates are obfuscated with an algorithm we do not specify further due to ethical considerations.

B↓ Bit→	0 31	32 63
0	date	patch ID
8	patch block | len | init	checksum
16	northbridge ID	southbridge ID
24	CPUID	magic value
32	match register 0	match register 1
40	match register 2	match register 3
48	match register 4	match register 5
54	match register 6	match register 7
64	triad 0, microinstruction 0	
72	triad 0, microinstruction 1	
80	triad 0, microinstruction 2	
88	triad 0, sequence word	triad 1 ...

Table 5.1: Microcode update file format.

Microcode Update Procedure. The microcode update binary is uploaded to the CPU in the following way: First, the patch must be placed in accessible virtual address space. Then the 64-bit virtual address must be written to MSR`0xc0010020`. Depending on the update size and microarchitecture, the `wrmsr` instruction initiating the update takes around 5,000 cycles to complete. A patch rejection causes a general protection fault. Internally, the update mechanism verifies the checksum, copies the triads to microcode patch RAM, and stores the match register fields in the actual match registers. Patch RAM is mapped into the address space of the microcode ROM, whereby the patch triads directly follow the read-only triads in the ROM structure.

Match Registers. The match registers are an integral part of the update mechanism. They hold a microcode ROM address, intercept the triad stored at that location, and redirect control to the triad in patch RAM at the offset *match register index* · *2*. A shared address space enables microcode in the patch RAM to jump back to microcode ROM, e.g., to reuse existing triads. Due to the complexity of the microcode update procedure, we assume it is implemented in microcode itself. We summarize our understanding of the microcode update mechanism in

Figure 5.1. AMD's patent [98] from 2002 describes an example microcode patch device and provides an idea of how the internals work.

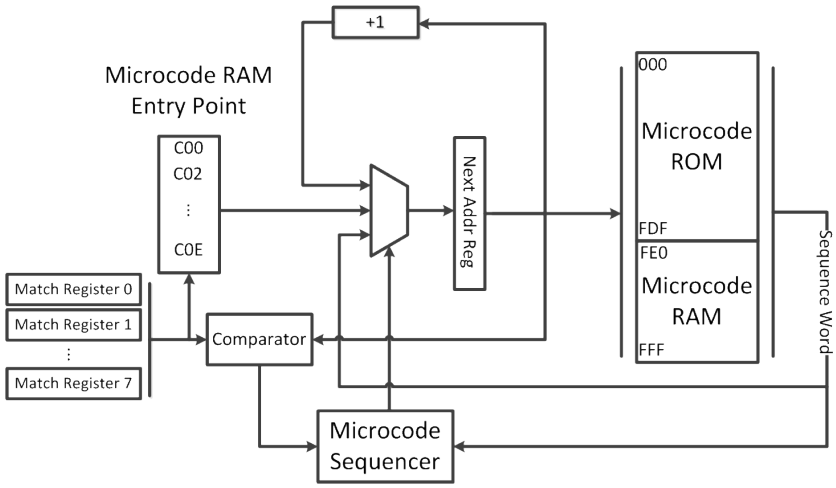


Figure 5.1: Overview of the AMD microcode update mechanism.

5.3.3 Framework/Low-Noise Environment

Since we did not have access to CPU internals, we had to be able to apply our extracted (and crafted) microcode updates and experimentally reverse engineer the CPU's behavior (e.g., register value changes and memory addresses). To pinpoint exactly where the changes caused effects (down to a single macroinstruction), we had to eliminate any *noise* from the operating system code execution or parallel threads out of our control. For example, common operating systems implement task switching or fully symmetric multiprocessing, which is undesirable in our setting.

The newly generated updates' code execution is capable of triggering abnormal behavior and then can most likely cause a system crash. Hence, we built a *low-noise* environment where we have full control of all code to realize accurate observation of the CPU state and behavior. It represents one minimal computer with an AMD CPU that runs our low-noise environment and is connected to a Raspberry Pi via a serial bus. To enable monitoring and control, the mainboard's power and reset switch, as well as the power supply's, are connected to GPIO ports. The Raspberry Pis run Linux and can be remotely controlled from the Internet. These development machines are used to design test cases and extend the microcode API. Furthermore, test cases can be launched from the development machines. This process automatically transfers the test case and the latest API version to the desired nodes(AMD CPUs), which then autonomously execute the test case and store the results. Our test setup consists of three nodes consisting of a K8 Sempron 3100+ (2004), K10 Athlon II X2 260 (2010), and K10 Athlon II X2 280 (2013) processor.

With the low-noise framework, we have successfully reverse engineered some parts of microcode format that allows us to write our own microcode updates. As the software part of the microcode is not the main contribution of this thesis, we refer the interested reader to [88]. Even though we had a basic understanding of the microcode, allowing us to modify and write arbitrarily,

some microcoded instructions remain unknown to us. They used CPU internal registers and were hard to interpret without code flow. Hence, we required the masked ROM and hardware mapping to get a better understanding of the microcode ISA. In the following, we'll concentrate on the hardware side of the AMD K8 mask ROM.

5.4 Hardware Reverse Engineering of Microcode ROM

In addition to microcode update mechanism, we reverse engineered the microcode in a black box scenario. The goal of the hardware analysis is to read and analyze the non-volatile microcode ROM to support reverse engineering of the microcode encoding. Furthermore, in combination with the physical mapping this allows to analyze the actual implementation of microcoded macroinstructions(vector path).

Our chosen DUT is a Sempron 3100+ (SDA3100AIP3AX) with a 130nm technology size since it features the largest size of the target CPU family (which facilitates our analysis). Note that the larger technology size allows for additional tolerance margins in both the delayering and the imaging of the individual structures. In contrast to traditional uC, general-purpose x86 CPUs feature a much larger die size and are stacked up to 12 layers, which increases hardware reverse engineering effort. This CPU is built in a SOI process.

We expected the targeted non-volatile microcode ROM to be stored in a cell array architecture. Other memory types to implement microcode ROM, such as flash, Electrically Erasable Programmable Read-only Memory (EEPROM), and RAM, are either too slow, unnecessarily large, or volatile.

In the following, we depackaged multiple samples from the metal heat sink with a drill. The underlying die is mounted as a flip-chip with the backside on top, revealing a silicon bulk on the backside. On the front side of the die is the PCB to connect the die with the BGA sockel. Hence, it is removed with fuming acid without damaging the chip protected by the passivation(SIO_2). As an unknown manufacturing technology, we expected multiple failed attempts during deprocessing. Therefore, multiple dies for different processing steps have been prepared ①-⑤. This allows multiple tries to adjust and fine-tune the equipment to acquire high quality prepared microcode mask ROM images.

5.4.1 Microcode masked ROM Overview

The first extracted die ① is thinned from the backside to get possible ROI locations from the visible functional blocks. Once the ROI location is known, the following deprocessing and analysis steps[85] facilitates our deprocessing. The backside Si bulk($> 150\mu m$) can be removed mechanically in a CNC polishing process or with wet chemicals e. g., TetraMethyl Ammonium Hydroxide (TMAH) and Potassium Hydroxide (KOH). In our case we chose to etch with TMAH due to the SOI fabrication process of the DUT. The comparison of default bulk processes and the SOI can be seen in Figure 5.2b. Briefly, the SOI process is attractive because it greatly reduces the source/drain diffusion capacitance, resulting in faster and power-efficient transistors[155]. The interested reader is referred to [155].

The ratio of etch rates between Si and SIO_2 is, in a low temperature environment, high enough to create a highly selective etch process[21]. This allows us to manually by inspecting the sample

(a) Si-bulk process (b) SOI process

Figure 5.2: Si-bulk process compared to SOI

every 10 minutes, without removing the SOI barrier. After etching for several hours, only the SiO_2 barrier and a very thin SI area under the gates is left. The thin remaining Si does not block visible light.

The result after the etching can be seen in Figure 5.3. Note that the general die structure is almost identical to the die image provided in [49] which helped in our initial analysis to identify our ROI: the microcode ROMs. Please note, that an overview can also be seen by using an infrared light above 1200 μm after depackaging. This reduces the preparation effort but sacrifices resolution due to the long wavelength.

We identified four ROM blocks (R1 to R4) consisting each of 8 subarrays. Prior to delayering the microcode memory structure of the AMD K8, we expected a ROM structure, as a highly optimized fast memory structure that is build in high density. This saves space on the die, resulting in a smaller die and a better yield overall being more economic during fabrication. Thus we expect *NAND* ROM or *NOR* ROM. Each introduced ROM can encode the content in different layers and therefore different layouts [131]. Nevertheless, at this point in time, we did not know the exact memory structure, so we utilize the thinned die ① in order to identify the memory structure in the following.

5.4.2 ROM structure and Memory Identification

The first die ① is further used to identify and understand the ROM structure. Therefore the CPU is fixed on sample holder and a hole of 5 μ by 5 μm is created with the FIB capabilities from the ultra thin backside. The sample can be seen in Figure C.2 in the Appendix.

During the process, starting from the backside, we can see every layer of the memory structure. As memory structures are regular in a big area. The exact location of inspection is not that critical and even allows failed attempts. As we are capable to navigate with the FIB, we arbitrarily chose one location. Note, that the mask ROM area is big enough for multiple attempts if one layer was mistakenly removed and even allows to reveal each layer in a different spot if desired. The (manually) vectorized images of the intermediate layers can be used to construct a 3D layout and identify the ROM memory type.

As can be seen in Figure C.2, we encountered a layout resembling Figure C.1 as regular NOR ROM arrays using the contact layer (vias) for programming. In NOR ROM with active layer programming, the logic state is encoded by the presence or absence of a transistor [131]. The

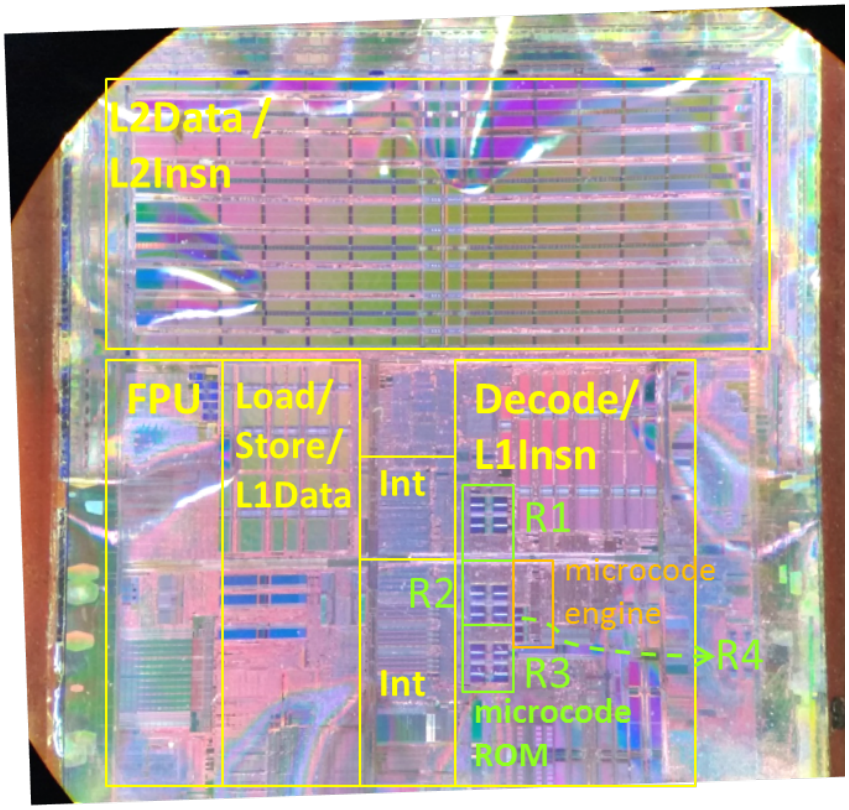


Figure 5.3: Die shot of AMD K8 Sempron 3100+ with different CPU parts. The image was taken with an optical microscope with low magnification. The die is corrugated due to a remaining thickness below 10 micrometers.

memory structure is altered for improved performance, but the basic structure from Figure C.1 is still applicable.

In this case, an advanced *bitline-folding* architecture [77] encodes the logic state by either placing a via on the right or the left bitline. The twisted bitline architecture can be seen in Figure 5.4 and in the bottom of Figure 5.6. This specific ROM may only have a single via set at any time; setting both will result in a short circuit. We used this characteristic to improve the image processing in Section 5.4.4.

5.4.3 ROM Acquisition

Further samples ②-⑤ are used to extract the ROM content by deprocessing. Multiple samples of the same CPU type were used to increase the quality and success chance, as the planar delayering is exponentially difficult with increasing die size and decreasing technology sizes [108, 85]. In order to visualize the ROM array, we *delayered* (e.g., removed individual stacked layers) from the top of the die. The main challenge during delayering is to uniformly skim planar surfaces parallel to the individual layers. Typically, the delayering process alternates between removing a layer and imaging the layer beneath it [108]. Focusing on our ROI, we were able to neglect other areas of the chip resulting in a more planar surface in the important region(s). Note that

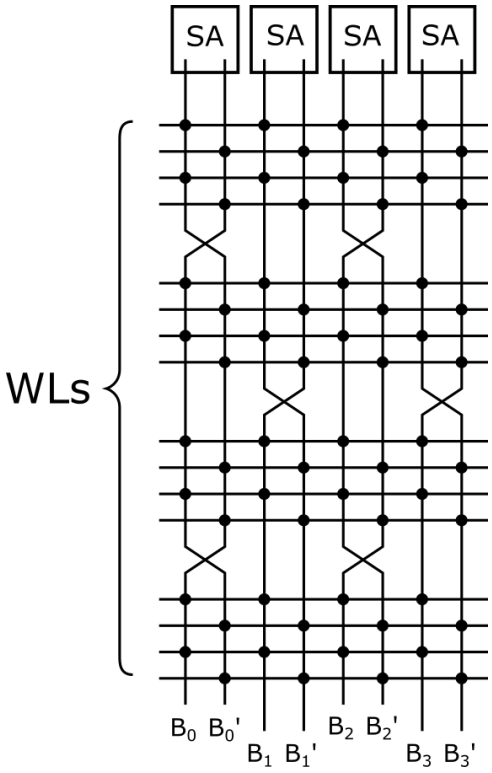


Figure 5.4: Transposed/Twisted bitline architecture reduces the capacitance between two bitlines. Sense Amplifiers(SA) amplify the difference of two bitlines.

hardware reverse engineering of the whole CPU architecture would require a more controlled delayering process and several months to acquire and process the whole layout. One full ROM region (R1) can be seen in Figure 5.5.

In order to remove layers, we used a combined approach of CMP and plasma etching. During inspection of the 7^{th} layer[2], we encountered the expected ROM array structure. We acquired images of individual layers using a SEM since optical microscopy reaches diffraction limits at this structure size. Compared to colored and more transparent images from optical microscopy, SEM images only provide a gray-scale channel, but with higher magnification. In SEM images, different materials can be identified due to brightness yield.

The physical storage is composed out of three larger regions of ROM (R1 to R3), which were later identified as the area containing the operations, and a smaller region (R4) containing the sequence words. Our work performs permutations such as inversion and interleaving of bit rows to receive whole operations in correct bit order. The R1 to R4 can be seen in Figure 5.3. In addition, the algorithm for constructing triads out of three operations is reverse engineered. The triads are built by loading a single operation out of each of the three regions R1 to R3 and loading the corresponding sequence word from region R4. Thereby the operations belonging to one triad have the same offset relative to the start of their corresponding region. The different

[2] counted from top to bottom

subregions of a single ROM region are illustrated in Figure 5.5. We will use the same naming convention in the following.

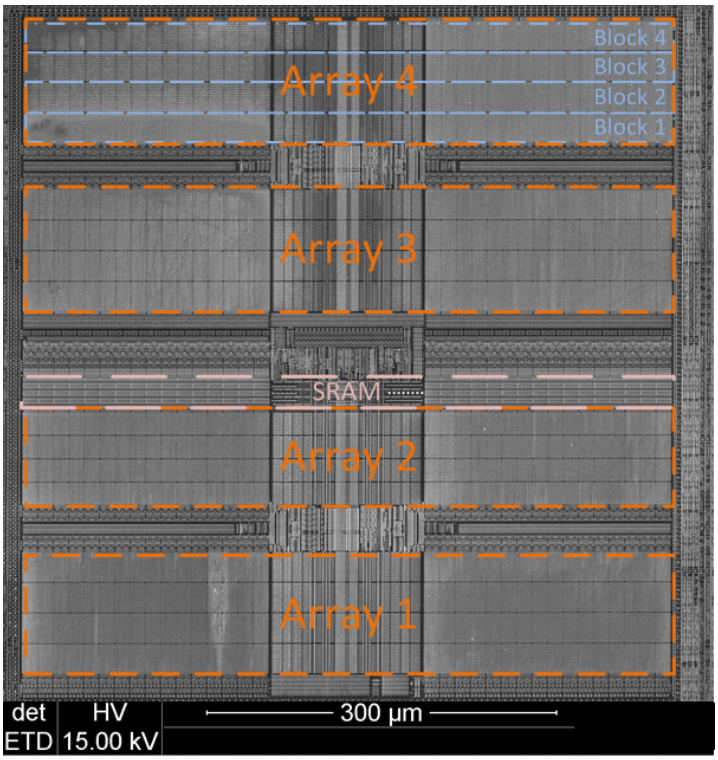


Figure 5.5: SEM image of region R1 showing arrays A1 to A4 and the SRAM holding the microcode update.

We were likewise able to locate the microcode patch RAM, which is loaded with the microcode updates during runtime. The RAM needs to be placed physically close to the rest of the microcode engine to keep signal paths short. The area between arrays A2 and A3 was classified as Static Random Access Memory (SRAM) and is marked in Figure 5.5. This location also contains a visually different control logic, which indicates a different type of storage than the rest of the region. It should be noted that the usage of two different classes of storage in this close proximity implies a highly optimized hardware layout, prior unknown in publications to the best of the authors' knowledge. The SRAM marked in the figure contains 32×64 bits, which is the size of data needed for 32 triads per region. This corresponds to the maximum update size of 32 triads determined in our software experiments. Due to the additional complexity of implementing a fast readable and writable memory in hardware, the SRAM occupies roughly the same space as a ROM block with 256 triads. This highlights the better density and hence the economic advantage of mask ROM.

With the knowledge of the triads size and structure from the former sections, the hardware layout suggested that the triads are organized in four arrays (A1 to A4) per region (R1 to R4), with A1, A3 and A4 containing data for 1024 triads each and corresponding 768 triads in A2 which is physically smaller than the other arrays. This organization means that the first triad, will use a microinstruction extracted from R1:A1, R2:A1 and R3:A1 as its operations and the

sequence word is obtained from R4:A1. As the regions are no longer relevant after combining the triads, they will be omitted in further notations.

Each of the arrays is subdivided into blocks B1 to B4, each containing 256 triads. The exception to this is the array A2; while the hardware layout suggests the presence of four blocks with a smaller number of triads each, we mapped the contents to three blocks with 256 triads each. This means array A2 contains only 768 triads in contrast to the 1024 triads contained in the other arrays.

5.4.4 Microcode Extraction

In Figure 5.6, we highlighted how bits are programmed by this memory type. Bright spots represent a via going down from a metal line, which is either connected to GND or VCC. We chose to represent the individual cells as set to logical '1' if the left via was set and '0' if the right one was set. This convention does not necessarily correspond to the correct runtime interpretation. However, permutations are commonly applied to the ROM memory, hence a misinterpretation can be corrected in a later analysis step in software.

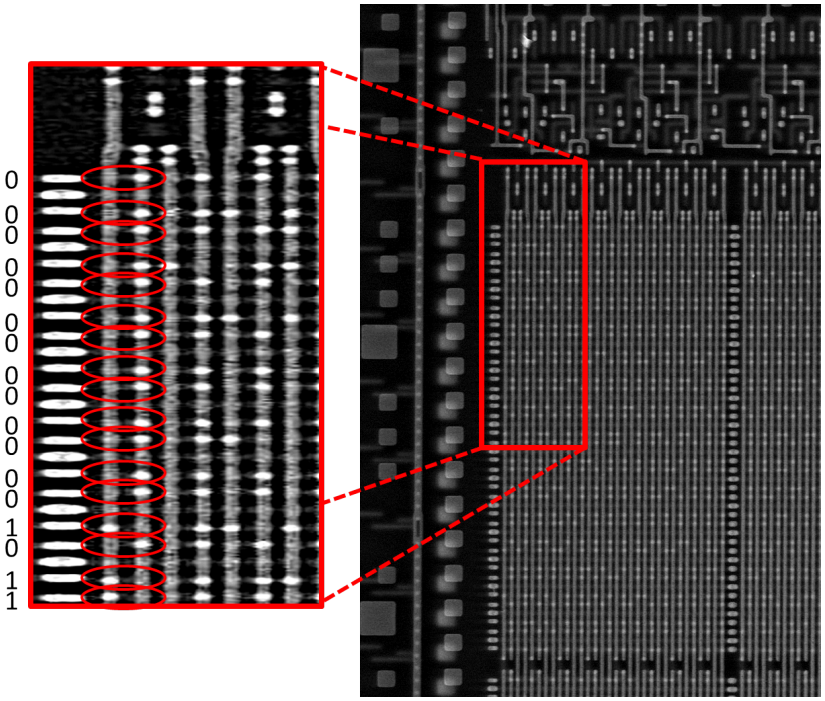


Figure 5.6: Partially interpreted bits in one ROM subarray.

In order to analyze the microcode ROM bits for any permutations, we processed the acquired SEM images with a slightly altered *rompar* script [13]. Using its image processing capabilities, we transformed the optical via positions into bit values.

Microcode ROM Bit Analysis. In order to group the bit values into microinstructions, we carefully analyzed the ROM structure and we made three crucial observations:

(1) Each alternating column of bits is inverted due to mirroring of existing cells. With a common GND line in between two cells, as seen in Figure C.1, the structure is more efficient.

(2) We identified that the ROM employs a transposed bitline architecture [77], that is often referred in combination with DRAM techniques. The twisted bitline architecture can be seen in Figure 5.4 and in the bottom of Figure 5.6. Therefore, the bit inversion has to be adjusted to each block.

(3) Each block from Figure 5.5 is separated into 32×16 bit-columns and 32 rows. One 16 bit-column, as seen in Figure 5.6, can be separated into 2×8 bit each contributing a single bit to a microinstruction.

With all three observations in mind, we were able to derive microinstructions from the images. In detail, on row consisting of $32 \times 16 = 32 \times 2 \times 8$ bit blocks generates 8 microinstructions by extracting one bit, always with the same offset, out of the 8-bit blocks. Hence, the ROM allows us to find more complex microinstructions and experimentally reverse engineer their meaning. Note, that this only results in multiple unrelated microinstructions, not necessarily consecutive ordered sequences.

Overall, we identified three ROM blocks (R1 to R3) consisting of 4 subarrays. Each of the 3 ROM blocks has the capability to store 30 kB. Note that our results match the visible blocks in [49]. Furthermore a small SRAM array was identified which can be seen in Figure 5.5, most likely being the microcode RAM. We would like to emphasize that this is a rather uncommon practice and is the first time we encounter a hybrid memory structure consisting of RAM and ROM. As SRAM is a volatile memory type it cannot be read-out optically. We want to remind that the vias positions are hardwired and cannot be changed after shipping. The only possible way to patch bugs in the ROM is to employ the microcode update procedure described in Section 5.3.2 with the shown SRAM.

5.5 Physical Mapping

So far the results are multiple unrelated microinstructions, not necessarily consecutive ordered sequences. By establishing a link between the addresses used in the microcode ROM and the physical layout present in the hardware, we try to order the microinstructions in order to read whole implementations of x86 instructions. We define two different classes of addresses:

- *virtual addresses* are used when the microcode software refers to a specific triad, e.g. in the match registers or jumps.
- *physical addresses* are the addresses assigned to triads after ROM readouts with initial interleaving and reordering.

Please note that the address granularity for microcode is one triad, the individual operations forming a triad are not addressable. Thus, it is our goal to reverse engineer the mapping of a given virtual address to its corresponding physical address. In the following, we had made an educated guess on an initial sequence from the hardware perspective. This initial sequence is in the following corrected by emulating different microcode sequences possibilities with a developed

microcode engine. This intermediate order, in combination with the initial hardware ordering results in the final mapping that we present in Section 5.5.2. Please note that this approach is also applicable to the similar architectures.

In the **initial ROM read-out** we chose an arbitrary sequence of the arrays A1 to A3 within one ROI and set A2 at the end to build a big sequence of zeros in the smallest array A2. This leads us to assume that A2 contains the highest addresses and the ROM is filled up with zeros. With the help of this observation, we have been able to cluster tailing zeros by an 8 microcode interleaving with two consecutive rows alternating. By testing and emulating different translations, we are able to identify the microinstruction order within the arrays in the following section.

5.5.1 Microcode Emulation

In this chapter, we have shown the possibility to reverse engineer the microcode content its update mechanism. The shown microcode ROM is a hard-wired NOR mask ROM that uses the contact layer for programming. The ROM structure found was a hybrid memory structure, mixing SRAM in the microcode ROM and share addresses decoders. Further techniques known from DRAM patents have been identified to improve the performance of the ROM. Around 100 kB ROM content has been retrieved and mapped to the virtual physical addresses within the microcode address space. With this, the implementation of vector path macro-instructions in the AMD K8 is known. This allows to analyze the microcode updates and gives a deep insight of the CPU internals working.

And last but not least in cooperation we are able to implement our own microcode instructions, in real-world AMD K8 CPUs. Malicious microcode trojans, defense mechanisms(e. g., for spectre) and utility functions have been shown with microcode updates in [88, 87].

5.5.2 Physical Ordering

The results from the emulation are hence applied to the physical images and therefore gain the physical order in the raw microcode ROM. Since the approach is in general the same for every array we describe it with A4 as an example. The first microinstruction is in row 0 with offset(i) 0 on the top-right side in Figure 5.7.

Please note that every row has exactly 8 microinstructions stored. Each block consists of 32 rows. Figure 5.7 shows A4 and the sequence in which order to extract the microinstructions. Each i_{th} opcode from a row is taken and the arrow to the next row followed. Each same-indexed instruction from two consecutive rows (which build a row-pair, e. g., r0/r1 and r2/r3 are two pairs) are alternatively used until the end of the block. This means in each block we iterate the rows in the following sequence: $[0, 1, 4, 5, 8, 9, ..., 28, 29]$ before increasing the index i and rerunning this sequence. Since each row has exactly 8 microinstructions, we retrieve 8×16 microinstructions in the correct order. Once these 16 rows are retrieved, the next block is read out in the same way. Please note that this appears like 16 rows per block are left behind. They are written in a left alignment in Figure 5.7 and will be read-out later in the process.

This sequence is repeated until row 29 from B4 is read. At this point, we identified the next microinstruction in B4 row 30/31 followed by row 26/27, again starting with the first indexed microcode($i = 0$). This is indicated on the left side of Figure 5.7. The sequence for each block on the left side is $[30, 31, 26, 27, 22, 23, ..., 2, 3]$. Furthermore, the flow goes backward from B4 to B1

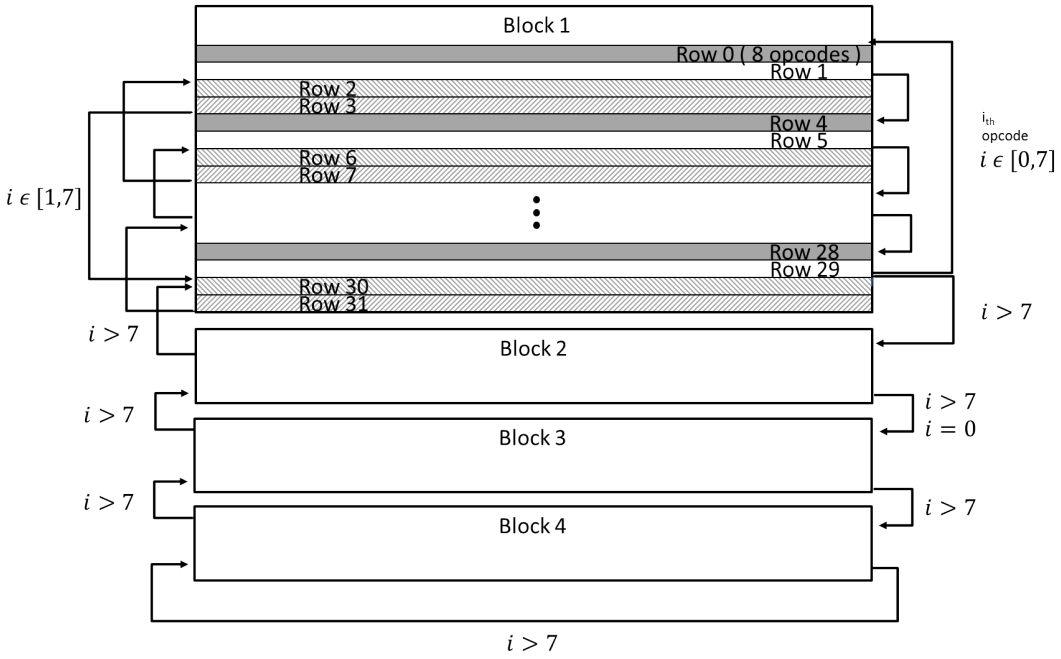


Figure 5.7: Reversed microinstruction mapping in the physical array A4. Beginning in the topright we acquire 1024 microinstructions with the sketched read-out.

until we reach B1 row 2,3. Again, after one block has reached row 2/3, the offset i is increased before jumping in the in next block. In this manner, we retrieve 1024 ordered microinstructions from array A4.

Array A1, A2, and A3 are similarily build to A4. The same approach can be applied with minor changes due to the efficiently designed hardware structures. In fact, A1 and A3 are inverted (in detail; mirrored along the horizontal axis) to use shared structures. Hence, the starting point is in the high rows 30,31 of block 4 and the direction is reversed towards lower indexed rows. Nevertheless, the round-like movement around one array stays the same.

Please note that the high-level regions R1 to R4 need to be interleaved to create a complete microcode triad. With a minimal sample extracted in the above-mentioned order, the internal order of the 3 microinstructions that build one triad is gained in a trivial manner due to the data flow of registers: R1-R3-R2-R4. Note that R4 is the ROI containing only sequence words and is set to be the last in any case. For R4, the above-described approach is the same, slightly altered to the smaller memory structure.

In the end, we have a continuous physical memory space with addresses starting at 0 and increasing with each triad to 0xEFF. This corresponded with the observation that the microcode patch RAM starts at the address 0xF00 for the K8 series of processors. This allows to 'disassemble' the microcode ROM and see the macro-instructions implementation in a custom build software.

5.6 Conclusion

In this chapter we have shown the possibility to reverse engineer the microcode content an its update mechanism. The shown microcode ROM is a hard-wired NOR mask ROM that uses the contact layer for programming. The ROM structure found was a hybrid memory structure, mixing SRAM in the microcode ROM and share addresses decoders. Further techniques known from DRAM patents have been identified to improve the performance of the ROM. Around 100 kB ROM content have been retrieved and mapped to the virtual physical addresses within the microcode address space. With this, the implementation of vector path macro-instructions in the AMD K8 are known. This allows to analyze the microcode updates and gives a deep insight of the CPU internals working.

And last but not least in cooperation we are able to implement our own microcode instructions, in real-world AMD K8 CPUs. Malicious microcode trojans, defense mechanisms(e. g., for spectre) and utility functions have been shown with microcode updates in [88, 87].

Chapter 6

Conclusion and Future Work

During the course of this thesis, we have dedicated our research efforts to the application of hardware reverse engineering. We challenge previous hardware reverse engineering with parametric hardware Trojans and utilize SCA to gain crucial gate-level netlist knowledge prior to deprocessing. The results are summarized below:

In Chapter 3 we revise an approach to pinpoint the ROI for ICs with VC images. This reduces the complexity of hardware reverse engineers by gaining apriori knowledge of the location of security-relevant ROI and allows us to perform EM-based SCA with a better signal-to-noise ratio. Furthermore, this enables more advanced SCAs like inter-gate leakages discussed in [137] and reduces the exhaustive search for fault attacks. We use the VC as a side channel that exploits the capacitive coupling effect of top metal layers through the covering passivation. We can see voltage alterations of the ICs surface in a SEM, which reveals secret information through top metal-wire voltage changes. Furthermore, we use the VC in multiple SCA approaches to recover the full AES key. The backside VC or EBP shown in [122] and [123] is implemented to read secret material. It shows a substantial hardware security threat, as we have shown that the backside thinning is applicable to large areas and it can be used in a SCA. Future work can highlight the threat of the backside VC on security microprocessors. If sufficient pinpointing backside ROI identification techniques, such as photon emission or Laser Voltage Probing (LVP), prove to be applicable, vendors and designers must respond with defensive techniques.

In Chapter 4, we leveraged parasitic effects of closely routed adjacent interconnects to design parametric malicious circuits, known as a crosstalk Trojan. Since such Trojans can only be realized by rerouting existing interconnections, they possess a zero gate overhead, which is both stealthy and difficult to identify with standard visual inspection techniques. In two case studies, we demonstrated that crosstalk Trojans provide reliable, adversary controllable faults to (i) insert malicious circuitry in a cryptographic AES IP to leak the secret key and (ii) elevate user access rights to allow arbitrary code execution in ring 0 for a modern and fully-fledged CPU. Since we believe that crosstalk Trojans are a building block of stealthy and hard-to-detect hardware Trojans, we recommend that the community include parametric characterizations in the hardware reverse engineering process to detect such advanced parametric Trojans. Because the crosstalk Trojan shown is proof of concept design, we believe an implementation with additional camouflaging techniques in a real chip shows further security threats.

In Chapter 5, we have shown the possibility to reverse engineer the microcode content and its update mechanism. The microcode ROM shown is a hard-wired NOR mask ROM that uses the contact layer for programming. Techniques known from RAM structures have been used to improve the performance of the ROM. Around 100 kB ROM content has been retrieved and

mapped to the virtual physical addresses within the microcode address space. In future work, the ROM content of similar proprietary microcode architectures can be analyzed to improve the performance and security of the system.

Hardware reverse engineering is a topic that includes many scientific areas that can be researched. We strongly believe it can be further improved by automation, robustness, and speed. As a result, the security community can use it in mid-cost approaches against malicious circuitry and hardware Trojans. The industry and academic community show increasing interest in this kind of engineering, based on the number of publications and dedicated devices on the market.

Part IV

Appendix

Appendix A

Voltage Contrast Side Channel Analysis Appendix

A.1 Additional Figures

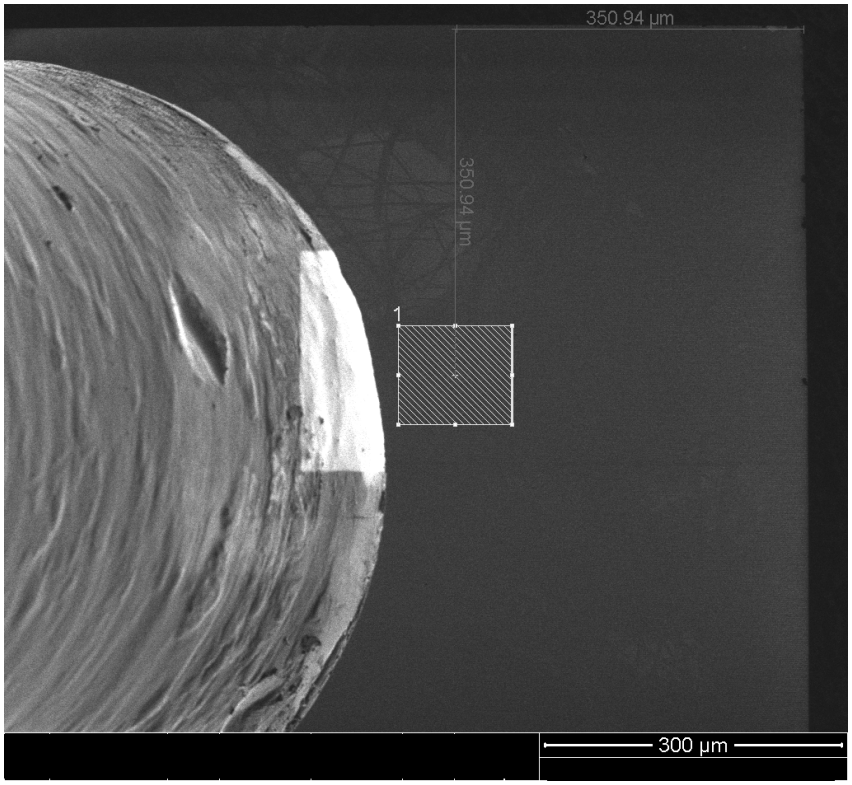


Figure A.1: The ROI from the backside. On the left side the gas-injection needle from the FIB can be seen.

Appendix B

Security Implications of Intentional Capacitive Crosstalk Appendix

B.1 Case Study: Cryptographic Designs

In the following we provide additional implementation specifics for the crosstalk Trojans used in case study on cryptographic designs described in Section 4.4. More precisely, we provide details on the resource utilization in Table B.1, Trojan specifics in Table B.2,

Design Status	Routed
Clock frequency of AES	$500MHz$
# Instances	25106
# Hard Macros	0
# Std Cells	25106
# Pads	0
# Net	11848
# Special Net	2
# IO Pins	31
# Pins	43944
# PG Pins	50212
Average Pins Per Net	3.709

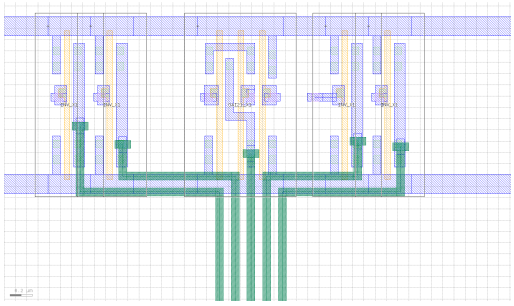
Table B.1: Excerpt of the general design information for AES IP core after place-and-route.

B.2 Case Study: CPUs

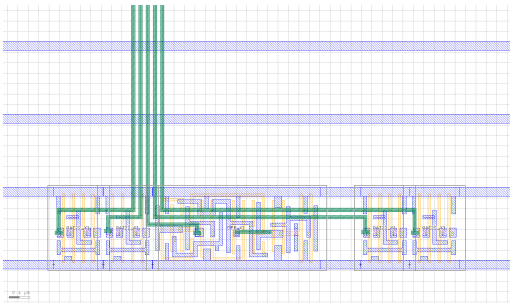
Table B.3 shows the final design summary.

last_ round DFF v_t	$0.54V$
Setup time of *last_ round* DFF	$0.06ns$
Hold time of *last_ round* DFF	$0.12ns$
Coupling length of Trojan	$1.25mm$
Separation between Trojan wires	$70nm$
Trojan metal layer	Metal 3
Crosstalk peak at victim net	$676.6mV$
Crosstalk width at victim net	$0.74ns$

Table B.2: Properties of the crosstalk Trojan within the AES core. The *last_ round* DFF is directly connected to the victim. Depending on this DFF the core starts to output the ciphertext. The crosstalk peak needs to be $676mV$ for the DFF to recognize the signal as logical high.The AES is clocked at $500Mhz$ with a $45nm$ technology size. Minimum DRC separation gap is 70 nm on Metal-3.



(a) The 5 driver cells right before the crosstalk Trojan. The 4 aggressor wires are strong inverters while the receiver is rather weak.



(b) The 5 receiving gate cells. The victim is connected to a DFF to store the peak for multiple clock cycles. This will result in a key leakage if the victim is faulted with the crosstalk Trojan.

Figure B.1: AES crosstalk Trojan cells fixed placed and routed a priori to the normal place and route. Source(a) and destination(b) are $1.25mm$ apart.

Design Status	Routed
# Instances	1588885
# Hard Macros	0
# Std Cells	1588885
# Pads	0
# Net	689858
# Special Net	2
#IO Pins	383
# Pins	2201664
# PG Pins	3177770
Average Pins Per Net	3.191

Table B.3: Excerpt of the general design information for OpenRISC 1200 IP core after place-and-route.

Appendix C

Microcode ROM

C.1 NOR ROM contact-layer

NOR* ROM encoded by the *contact-layer mask is implemented by connecting the Metal-1 to diffusion by contacts(vias). It can be seen in Figure C.1. When the wordline (W_x) is pulled high, the corresponding bitlines are pulled to GND, if an transistor is existent by contact. Compared to *NOR* mask ROM in the active/implant area, this step is done later in the fabrication. This implements multiple transistors, but only the programmed ones representing a digital '0' are connected. The size of one cell is $11\lambda \times 7\lambda$.

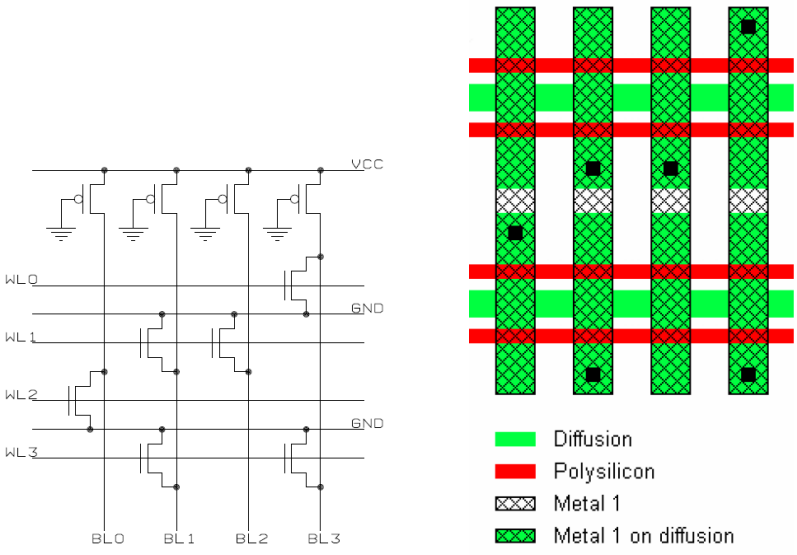
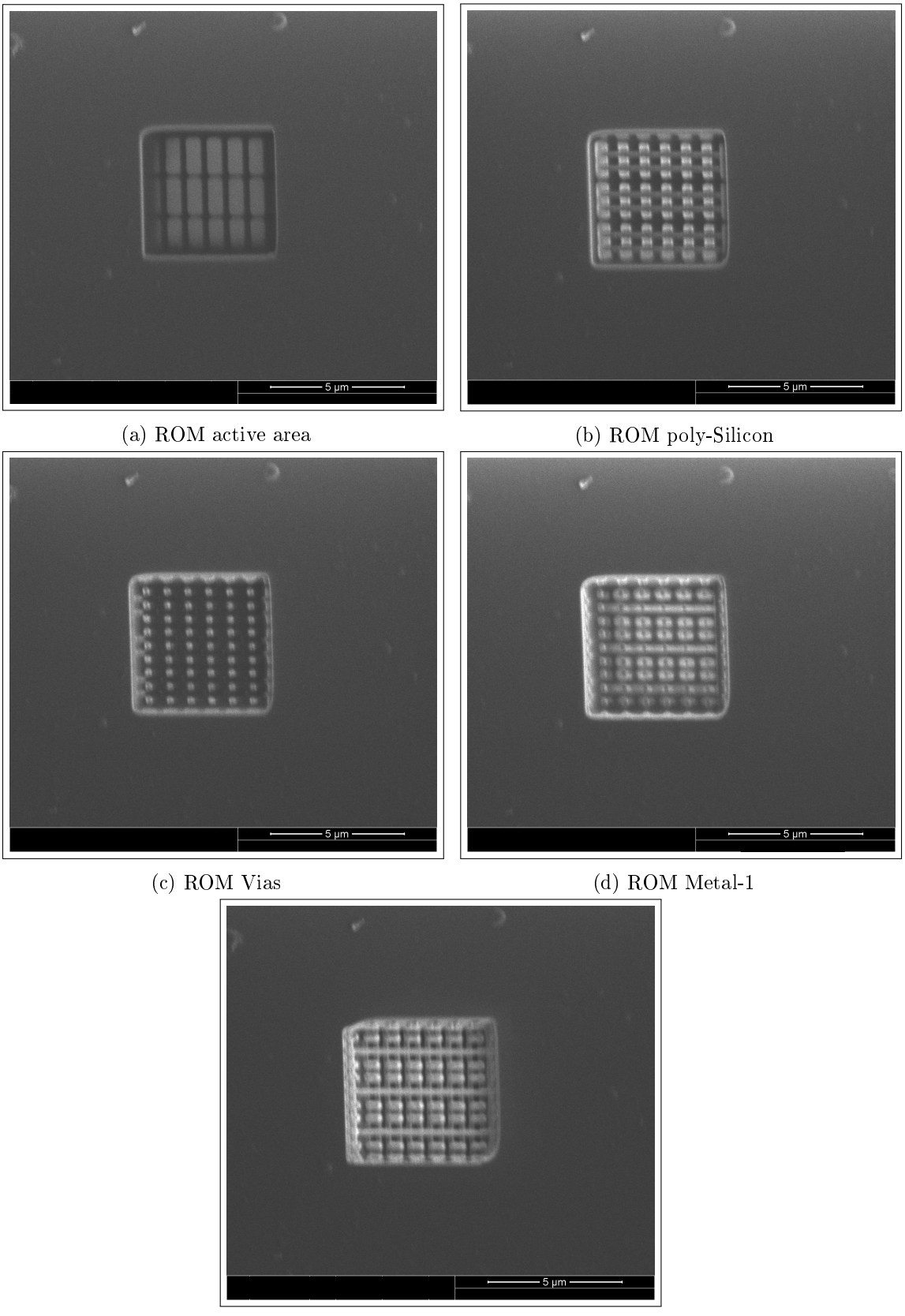


Figure C.1: NOR Mask ROM by Contact layer from [131]

C.2 Microcode ROM FIB Images



(a) ROM active area

(b) ROM poly-Silicon

(c) ROM Vias

(d) ROM Metal-1

(e) ROM within metal-1 and bitlines in the next layer can be seen

Figure C.2: Different ROM images. Images taken in a dual-beam FIB. These layer stacked resemble the NOR ROM in Figure C.1

Bibliography

[1] Image Decode Project. Webpage. [Online]. Available: `http://www.progettoemma.net/dump/`.

[2] Silicon Pr0n rom:mask. `https://siliconpr0n.org/wiki/doku.php?id=rom:mask`.

[3] Defense Science Board Whasington DC. Report of the Defense Science Board Task Force on High Performance Microchip supply, 2005.

[4] A. Vijayakumar . Physical Design Obfuscation of Hardware: A Comprehensive Investigation of Device and Logic-Level Techniques. *IEEE Trans. Information Forensics and Security*, 12(1):64–77, 2017.

[5] Advanced Micro Devices, Inc. AMD Athlon® Processor Model 10 Revision Guide, 2003.

[6] Advanced Micro Devices, Inc. AMD64 Virtualization Codenamed "Pacifica" Technology - Secure Virtual Machine Architecture Reference Manual, 2005.

[7] Advanced Micro Devices, Inc. Revision Guide for AMD Family 16h Models 00h-0Fh Processors, 2013.

[8] Ashok K. Agrawala and Tomlinson G. Rauscher. *Foundations of Microprogramming : Architecture, Software, and Applications*. Academic Press, 1976.

[9] Yousra Alkabani and Farinaz Koushanfar. Active Hardware Metering for Intellectual Property Protection and Security. In *USENIX Security Symposium*, 2007.

[10] Frederic Amiel, Benoit Feix, and Karine Villegas. Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms. In *Selected Areas in Cryptography*, pages 110–125, 2007.

[11] Anonymous. Opteron Exposed: Reverse Engineering AMD K8 Microcode Updates. [Online]. Available: `http://www.securiteam.com/securityreviews/5FP0M1PDF0.html`, 2004.

[12] ApertureLabsLtd. Fun with Masked ROMs - Atmel MARC4. `http://adamsblog.aperturelabs.com/2013/01/fun-with-masked-roms.html`.

[13] ApertureLabsLtd. Semi-automatic extraction of data from microscopic images of Masked ROM. `https://github.com/ApertureLabsLtd/rompar`.

[14] ARM, Inc. ARM Security Technology - Building a Secure System using TrustZone® Technology. [Online]. Available: `http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29GENC-009492C_trustzone_security_whitepaper.pdf`, 2005.

[15] Giuseppe Ateniese, Aggelos Kiayias, Bernardo Magri, Yiannis Tselekounis, and Daniele Venturi. Secure outsourcing of circuit manufacturing. Cryptology ePrint Archive, Report 2016/527, 2016. https://eprint.iacr.org/2016/527.

[16] Atmel. Atmel AVR XMEGA AU Manual, 04 2013.

[17] Todd M. Austin. DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design. In *Proceedings of IEEE/ACM International Symposium on Microarchitecture, MICRO 32*, pages 196–207, 1999.

[18] R. Barille. Analytical formulation of the capacitive coupling voltage contrast of a buried line. *Electronics Letters*, 29(20):1756–1758, Sept 1993.

[19] Daniel L Barton and Paiboon Tangyunyong. Thermal defect detection techniques. *Microelectronics failure analysis: desk reference*, page 378, 2004.

[20] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. Contactless electromagnetic active attack on ring oscillator based true random number generator. In Schindler, Werner and Huss, SorinA., editor, *Constructive Side-Channel Analysis and Secure Design*, volume 7275 of *Lecture Notes in Computer Science*, pages 151–166. Springer Berlin Heidelberg, 2012.

[21] Friedrich Beck. *Integrated Circuit Failure Analysis: A Guide to Preparation Techniques*. Wiley, 1998.

[22] Georg T. Becker, Francesco Regazzoni, Christof Paar, and Wayne P. Burleson. *Stealthy Dopant-Level Hardware Trojans*, pages 197–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[23] Jack D. Benzel. Bugs in Black and White: Imaging IC Logic Levels with Voltage Contrast. *HEWLETT PACKARD JOURNAL*, 46:102–106, 1995. 00000.

[24] Kirk J. Bertsche and Jr. Charles, H.K. The Practical Implementation of Voltage Contrast as a Diagnostic Tool. In *Reliability Physics Symposium, 1982. 20th Annual*, pages 167–178, March 1982.

[25] Shivam Bhasin and Debdeep Mukhopadhyay. Fault injection attacks: Attack methodologies, injection techniques and protection mechanisms. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 415–418, Cham, 2016. Springer International Publishing.

[26] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan. Hardware trojan attacks: Threat analysis and countermeasures. *Proceedings of the IEEE*, 102(8):1229–1247, Aug 2014.

[27] Eli Biham, Yaniv Carmeli, and Adi Shamir. Bug Attacks. In *CRYPTO*, pages 221–240, 2008.

[28] J.B. Bindell and J.N. McGinn. Voltage Contrast SEM Observations with Microprocessor Controlled Device Timing. In *Reliability Physics Symposium, 1980. 18th Annual*, pages 55–58, April 1980.

[29] Andrey Bogdanov and Takanori Isobe. How secure is aes under leakage. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, pages 361–385, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[30] Christian Boit, Clemens Helfmeier, and Uwe Kerst. Security Risks Posed by Modern IC Debug and Diagnosis Tools. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*, pages 3–11. IEEE, 2013.

[31] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure: SGX cache attacks are practical. In *USENIX Workshop on Offensive Technologies (WOOT)*, 2017.

[32] O Breitenstein, C Schmidt, and D Karg. Thermal failure analysis by IR lock-in thermography. *Microelectronics failure analysis desk reference, 5th ed., EDFAS*, 2004.

[33] Sébastien Briais, Stéphane Caron, Jean-Michel Cioranesco, Jean-Luc Danger, Sylvain Guilley, Jacques-Henri Jourdan, Arthur Milchior, David Naccache, and Thibault Porteboeuf. 3D hardware canaries. In *Cryptographic Hardware and Embedded Systems–CHES 2012*, pages 1–22. Springer, 2012.

[34] Carey Robertson. Solving the next parasitic extraction challenge. `http://www.techdesignforums.com/practice/technique/solving-the-next-parasitic-extraction-challenge/`, May 2018.

[35] Rajat Subhra Chakraborty and Swarup Bhunia. HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 28(10):1493–1502, 2009.

[36] George A Reis Jonathan Chang, David I August, and Robert Cohn Shubhendu S Mukherjee. Configurable Transient Fault Detection via Dynamic Binary Translation. In *Workshop on Architectural Reliability*, 2006.

[37] Daming D. Chen and Gail-Joon Ahn. Security Analysis of x86 Processor Microcode. [Online]. Available: `https://www.dcddcc.com/docs/2014_paper_microcode.pdf`, 2014.

[38] G. H. Chisholm, R. L. Veroff, S. T. Eckmann, and C. M. Lain. Understanding integrated circuits. *IEEE Design & Test of Computers*, 16:26–37, 04 1999.

[39] Robert Chivas and Scott Silverman Michael DiBattista. Adaptive grinding and polishing of silicon integrated circuits to ultra-thin remaining thickness, 11 2015.

[40] Christophe Clavier. An Improved SCARE Cryptanalysis Against a Secret A3/A8 GSM Algorithm. In *Information Systems Security*, pages 143–155. Springer-Verlag, 2007.

[41] Tim Coe. Inside the Pentium FDIV bug. Dr. Dobb's Journal of Software Tools, 20(4) , April 1995.

[42] Jason Cong, David Zhigang Pan, and Prasanna V. Srinivas. Improved crosstalk modeling for noise constrained interconnect optimization. In *Proceedings of the 2001 Asia and South Pacific Design Automation Conference*, ASP-DAC '01, pages 373–378, New York, NY, USA, 2001. ACM.

[43] Marc L Corliss, E Christopher Lewis, and Amir Roth. DISE: A Programmable Macro Engine for Customizing Applications. In *International Symposium on Computer Architecture*, pages 362–373, 2003.

[44] Victor Costan and Srinivas Devadas. Intel SGX Explained. Cryptology ePrint Archive, Report 2016/086, 2016. [Online]. Available: `http://eprint.iacr.org/2016/086`.

[45] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal Hardware Extensions for Strong Software Isolation. In *USENIX Security Symposium*, pages 857–874, 2016.

[46] Olivier Crépel, Felix Beaudoin, Lionel Dantas de Morais, Gérald Haller, C. Goupil, Philippe Perdu, Romain Desplats, and Dean Lewis. Backside hot spot detection using liquid crystal microscopy. *Microelectronics Reliability*, 42(9-11):1741–1746, 2002.

[47] R. Daudigny, H. Ledig, F. Muller, and F. Valette. SCARE of the DES. In *Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 393–406, 2005.

[48] Theo de Raadt. Intel Core 2. openbsd-misc mailing list. [Online]. Available: `http://marc.info/?l-openbsd-isc&m=118296441702631`, 2007.

[49] de Vries. Understanding the detailed Architecture of AMD's 64 bit Core. `http://www.chip-architect.com/news/2003_09_21_Detailed_Architecture_of_AMDs_64bit_Core.html`.

[50] Loïc Duflot. CPU Bugs, CPU Backdoors and Consequences on Security. In *ESORICS*, pages 580–599, 2008.

[51] Roger Durà, Jofre Pallarès, Raúl Quijada, Xavier Formatjé, Salvador Hidalgo, and Francisco Serra-Graells. Fast and robust topology-based logic gate identification for automated ic reverse engineering. In *ISTFA 2017*, 2017.

[52] Stefan Dziembowski, Sebastian Faust, and François-Xavier Standaert. Private circuits iii: Hardware trojan-resilience via testing amplification. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 142–153, New York, NY, USA, 2016. ACM.

[53] Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. Building a side channel based disassembler. In *Transactions on computational science X*, pages 78–99. Springer-Verlag, 2010.

[54] W.C. Elmore, Los Alamos Scientific Laboratory, and U.S. Atomic Energy Commission. *Transient response of damped linear network with particular regard to wideband amplifiers.* Atomic Energy Commission, 1947.

[55] Maik Ender, Samaneh Ghandali, Amir Moradi, and Christof Paar. The first thorough side-channel hardware trojan. Cryptology ePrint Archive, Report 2017/865, 2017. `http://eprint.iacr.org/2017/865`.

[56] FastFieldSolvers. FasterCap - Capacitance extraction at scale, June, 2018. [Online] Available: `https://www.fastfieldsolvers.com/fastercap.htm`.

[57] John G. Favor. Risc86 instruction set, January 1 2002. US Patent 6,336,178.

[58] J. Ferrigno and M. Hlavac. When AES blinks: introducing optical side channel. *Information Security, IET*, 2(3):94–98, September 2008.

[59] Forte, Domenic and Bhunia, Swarup and Tehranipoor, Mark M. *Hardware Protection through Obfuscation*. Springer, 1 edition, 2017.

[60] Mike Fournigault, Pierre-Yvan Liardet, Yannick Teglia, Alain Trémeau, and F. Robert-Inacio. Reverse Engineering of Embedded Software Using Syntactic Pattern Recognition. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 527–536, 2006.

[61] M. Fyrbiak, S. Wallat, P. Swierczynski, M. Hoffmann, S. Hoppach, M. Wilhelm, T. Weidlich, R. Tessier, and C. Paar. HAL—the missing piece of the puzzle for hardware reverse engineering, trojan detection and insertion. *IEEE Transactions on Dependable and Secure Computing*, 2018, to appear.

[62] Marc Fyrbiak, Sebastian Strauss, Christian Kison, Sebastian Wallat, Malte Elson, Nikol Rummel, and Christof Paar. Hardware reverse engineering: Overview and open challenges. In *IEEE 2nd International Verification and Security Workshop, IVSW 2017, Thessaloniki, Greece, July 3-5, 2017*, pages 88–94, 2017.

[63] Ravikishore Gandikota. Crosstalk noise analysis for nano-meter vlsi circuits. 01 2009.

[64] Adrià Gascón, Pramod Subramanyan, Bruno Dutertre, Ashish Tiwari, Dejan Jovanović, and Sharad Malik. Template-based circuit understanding. In *Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design*, FMCAD '14, pages 17:83–17:90, Austin, TX, 2014. FMCAD Inc.

[65] Gbur, Jerzy. Advanced Encryption Standard (AES), 2006. `https://opencores.org/project,aes_128_192_256,overview`.

[66] Samaneh Ghandali, Georg T. Becker, Daniel Holcomb, and Christof Paar. *A Design Methodology for Stealthy Parametric Trojans and Its Application to Bug Attacks*, pages 625–647. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[67] Ilias Giechaskiel and Ken Eguro. Information leakage between FPGA long wires. *CoRR*, abs/1611.08882, 2016.

[68] Sylvain Guilley, Laurent Sauvage, Julien Micolod, Denis Réal, and F. Valette. Defeating Any Secret Cryptography with SCARE Attacks. In *Progress in Cryptology - LATINCRYPT 2010* , pages 273–293. Springer-Verlag, 2010.

[69] Stanley Habib. Microprogrammed Enhancements to Higher Level Languages - an Overview. In *Workshop on Microprogramming*, pages 80–84, 1974.

[70] S. K. Haider, C. Jin, M. Ahmad, D. Shila, O. Khan, and M. van Dijk. Advancing the state-of-the-art in hardware trojans detection. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2017.

[71] M. C. Hansen, H. Yalcin, and J. P. Hayes. Unveiling the iscas-85 benchmarks: a case study in reverse engineering. *IEEE Design Test of Computers*, 16(3):72–80, 1999.

[72] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *2010 IEEE Symposium on Security and Privacy*, pages 159–172, May 2010.

[73] Jann Horn. Project zero: Reading privileged memory with a side-channel. [Online]. Available: `https://googleprojectzero.blogspot.co.at/2018/01/reading-privileged-memory-with-side.html`, 2018.

[74] Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh Tripunitara. Securing computer hardware using 3d integrated circuit (IC) technology and split manufacturing for obfuscation. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 495–510, Washington, D.C., 2013. USENIX.

[75] Intel Corporation. 6th Generation Intel® Processor Family Specification Update, 2016.

[76] Intel Corporation. Pentium® Pro Processor Specification Update, 2016.

[77] Bruce Jacob, Spencer Ng, and David Wang. *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann Publishers Inc., 2007.

[78] JohnDMcMaster. Silicon Pr0n tools. `https://github.com/JohnDMcMaster/pr0ntools`.

[79] Louise H Jones. A Survey of Current Work in Microprogramming. *Computer*, 8(8):33–38, August 1975.

[80] Andrew Kahng, Sudhakar Muddu, and Devendra Vidhani. Noise and delay estimation for coupled rc interconnects. In *In IEEE AISC/SoC*, 1999.

[81] Markus Kammerstetter, Markus Muellner, Daniel Burian, Christian Platzer, and Wolfgang Kastner. Breaking Integrated Circuit Device Security Through Test Mode Silicon Reverse Engineering. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 549–557, New York, NY, USA, 2014. ACM.

[82] Shahrzad Keshavarz, Falk Schellenberg, Bastian Richter, Christof Paar, and Daniel Holcomb. Sat-based reverse engineering of gate-level schematics using fault injection and probing. *CoRR*, abs/1802.08916, 2018.

[83] Samuel T. King, Joseph Tucek, Anthony Cozzie, Chris Grier, Weihang Jiang, and Yuanyuan Zhou. Designing and implementing malicious hardware. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, LEET'08, pages 5:1–5:8, Berkeley, CA, USA, 2008. USENIX Association.

[84] C. Kison, O. M. Awad, M. Fyrbiak, and C. Paar. Security implications of intentional capacitive crosstalk. *IEEE Transactions on Information Forensics and Security*, pages 1–1, 2019.

[85] Christian Kison, Jürgen Frinken, and Christof Paar. Finding the aes bits in the haystack: Reverse engineering and sca using voltage contrast. In *CHES*, pages 641–660. Springer, 2015.

[86] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *ArXiv e-prints*, 2018.

[87] Benjamin Kollenda, Marc Fyrbiak, Christian Kison Philipp Koppe, Christof Paar, and Thorsten Holz. Pending constructive microcode: Supplementing systems defenses with microcode. In *CCCS 2018)*, Toronto, Canada, 2018. ACM.

[88] Philipp Koppe, Benjamin Kollenda, Marc Fyrbiak, Christian Kison, Robert Gawlik, Christof Paar, and Thorsten Holz. Reverse engineering x86 processor microcode. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1163–1180, Vancouver, BC, 2017. USENIX Association.

[89] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian. Parametric trojans for fault-injection attacks on cryptographic hardware. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 18–28, Sept 2014.

[90] D. Lagunovsky, S. Ablameyko, and M. Kutas. Recognition of integrated circuit images in reverse engineering. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, volume 2, pages 1640–1642 vol.2, Aug 1998.

[91] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. Inferring fine-grained control flow inside sgx enclaves with branch shadowing. In *USENIX Security Symposium*, pages 16–18, 2017.

[92] W. Li, Z. Wasson, and S. A. Seshia. Reverse engineering circuits using behavioral pattern mining. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 83–88, June 2012.

[93] Wenchao Li, Adrià Gascón, Pramod Subramanyan, Wei Yang Tan, Ashish Tiwari, Sharad Malik, Natarajan Shankar, and Sanjit A. Seshia. Wordrev: Finding word-level structures in a sea of bit-level gates. In *HOST*, pages 67–74, 06 2013.

[94] Lang Lin, Markus Kasper, Tim Güneysu, Christof Paar, and Wayne Burleson. Trojan side-channels: Lightweight hardware trojans through side-channel engineering. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 382–395, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[95] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *ArXiv e-prints*, 2018.

[96] M. Hicks et.al. SPECS: A Lightweight Runtime Mechanism for Protecting Software from Security-Critical Processor Bugs. In *ASPLOS*, pages 517–529, 2015.

[97] Martin Maas, Eric Love, Emil Stefanov, Mohit Tiwari, Elaine Shi, Krste Asanovic, John Kubiatowicz, and Dawn Song. PHANTOM: practical oblivious computation in a secure processor. In *CCS*, pages 311–324, 2013.

[98] Kevin J. McGrath and James K. Pickett. Microcode patch device, August 27 2002. US Patent 6,438,664.

[99] T. Meade, Y. Jin, M. Tehranipoor, and S. Zhang. Gate-level netlist reverse engineering for hardware security: Control logic register identification. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1334–1337, May 2016.

[100] T. Meade, S. Zhang, and Y. Jin. Netlist reverse engineering for high-level functionality reconstruction. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 655–660, Jan 2016.

[101] Albert Meixner and Daniel J. Sorin. Detouring: Translating Software to Circumvent Hard Faults in Simple Cores. In *IEEE/IFIP International Conference on Dependable Systems and Networks, DSN*, pages 80–89, 2008.

[102] Stephen W. Melvin and Yale N. Patt. SPAM: A Microcode Based Tool for Tracing Operating Sytsem Events. *SIGMICRO Newsl.*, 19(1-2):58–59, June 1988.

[103] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa. Generation of performance constraints for layout. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(8):860–874, Aug 1989.

[104] Nangate. Freepdk45nm process design kit. `https://www.eda.ncsu.edu/wiki/FreePDK45:Contents`, August 2011.

[105] Karsten Nohl, David Evans, Starbug Starbug, and Henryk Plötz. Reverse-engineering a cryptographic rfid tag. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 185–193, Berkeley, CA, USA, 2008. USENIX Association.

[106] Roman Novak. Side-Channel Based Reverse Engineering of Secret Algorithms. In *Proceedings of the Twelfth International Electrotechnical and Computer Science Conference* , (ERK 2003), pages 25–26, 2003.

[107] OpenCores. Openrisc 1200 ip core specifcation. `https://www.isy.liu.se/en/edu/kurs/TSEA44/OpenRISC/or1200_spec.pdf`, 2001.

[108] Shahed E. Quadir, Junlin Chen, Domenic Forte, Navid Asadizanjani, Sina Shahbazmohamadi, Lei Wang, John Chandy, and Mark Tehranipoor. A Survey on Chip to System Reverse Engineering. *J. Emerg. Technol. Comput. Syst.*, 13(1):6:1–6:34, April 2016.

[109] J. Rajendran, V. Jyothi, and R. Karri. Blue team red team approach to hardware trust assessment. In *2011 IEEE 29th International Conference on Computer Design (ICCD)*, pages 285–288, Oct 2011.

[110] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. Security Analysis of Integrated Circuit Camouflaging. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 709–720, New York, NY, USA, 2013. ACM.

[111] Chethan Ramesh, Shivukumar B. Patil, Siva Nishok Dhanuskodi, George Provelengios, Sébastien Pillement, Daniel Holcomb, and Russell Tessier. FPGA Side Channel Attacks without Physical Access. In *International Symposium on Field-Programmable Custom Computing Machines*, page paper#116, Boulder, United States, April 2018.

[112] Tomlinson Gene Rauscher and Phillip M. Adams. Microprogramming: A Tutorial and Survey of Recent Developments. *IEEE Trans. Computers*, 29(1):2–20, 1980.

[113] D. Real, F. Valette, and M. Drissi. Enhancing correlation electromagnetic attack using planar near-field cartography. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 628–633, April 2009.

[114] Chris Richardson. Contoured Device Sample Preparation for 5 um Remaining Silicon Thickness (RST) Tolerances, 2014.

[115] Ruediger Rosenkranz. Failure localization with active and passive voltage contrast in FIB and SEM. *Journal of Materials Science: Materials in Electronics*, 22(10):1523–1535, 2011.

[116] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. A Primer on Hardware Security: Models, Methods, and Metrics. *Proceedings of the IEEE*, 102(8):1283–1295, 2014.

[117] Joanna Rutkowska. Intel x86 considered harmful. [Online]. Available: `https://blog.invisiblethings.org/2015/10/27/x86_harmful.html`, 2015.

[118] S. Narayanasamy et. al. Patching Processor Design Errors. In *International Conference on Computer Design ICCD*, pages 491–498, 2006.

[119] Smruti Sarangi, Satish Narayanasamy, Bruce Carneal, Abhishek Tiwari, Brad Calder, and Josep Torrellas. Patching Processor Design Errors with Programmable Hardware. *IEEE Micro*, 27(1):12–25, 2007.

[120] L. Sauvage, S. Guilley, J.-L. Danger, N. Homma, and Y. Hayashi. Practical results of EM cartography on a FPGA-based RSA hardware implementation. In *Electromagnetic Compatibility (EMC), 2011 IEEE International Symposium on*, pages 768–772, Aug 2011.

[121] Laurent Sauvage, Sylvain Guilley, and Yves Mathieu. Electromagnetic Radiations of FPGAs: High Spatial Resolution Cartography and Attack on a Cryptographic Module. *ACM Trans. Reconfigurable Technol. Syst.*, 2(1):4:1–4:24, March 2009.

[122] Rudolf Schlangen, R Leihkauf, U Kerst, Christian Boit, and Bjorn Kruger. Functional IC analysis through chip backside with nano scale resolution-E-beam probing in FIB trenches to STI level. In *Physical and Failure Analysis of Integrated Circuits, 2007. IPFA 2007. 14th International Symposium on the*. IEEE, 2007.

[123] Rudolf Schlangen, Reiner Leihkauf, Uwe Kerst, Christian Boit, Rajesh Jain, Tahir Malik, K Wilsher, Ted Lundquist, and Bjorn Kruger. Backside e-beam probing on nano scale devices. In *2007 IEEE International Test Conference*, 2007.

[124] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple Photonic Emission Analysis of AES. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 41–57. Springer Berlin Heidelberg, 2012.

[125] Kenneth L. Shepard and Vinod Narayanan. Noise in deep submicron digital design. In *Proceedings of the 1996 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '96, pages 524–531, Washington, DC, USA, 1996. IEEE Computer Society.

[126] Y. Shi, B. H. Gwee, Ye Ren, Thet Khaing Phone, and Chan Wai Ting. Extracting functional modules from flattened gate-level netlist. In *2012 International Symposium on Communications and Information Technologies (ISCIT)*, pages 538–543, Oct 2012.

[127] Y. Shi, C. W. Ting, B. H. Gwee, and Y. Ren. A highly efficient method for extracting fsms from flattened gate-level netlist. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 2610–2613, May 2010.

[128] Y. Shiyanovskii, F. Wolff, A. Rajendran, C. Papachristou, D. Weyer, and W. Clay. Process reliability based trojans through nbti and hci effects. In *2010 NASA/ESA Conference on Adaptive Hardware and Systems*, pages 215–222, June 2010.

[129] S. Skorobogatov. Using Optical Emission Analysis for Estimating Contribution to Power Analysis. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2009 Workshop on*, pages 111–119, Sept 2009.

[130] Sergei Skorobogatov and Christopher Woods. Breakthrough silicon scanning discovers backdoor in military chip. In *Proceedings of the 14th international conference on Cryptographic Hardware and Embedded Systems*, CHES'12, pages 23–40, Berlin, Heidelberg, 2012. Springer-Verlag.

[131] Sergei P. Skorobogatov. *Semi-Invasive Attacks – A New Approach to Hardware Security Analysis*. PhD thesis, University of Cambridge, 2005.

[132] William Stallings. *Computer Organization and Architecture: Designing for Performance (7th Edition)*. Prentice-Hall, Inc., 2005.

[133] Daehyun Strobel, Benedikt Driessen, Timo Kasper, Gregor Leander, David Oswald, Falk Schellenberg, and Christof Paar. Fuming acid and cryptanalysis: Handy tools for overcoming a digital locking and access control system. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 147–164. Springer Berlin Heidelberg, 2013.

[134] P. Subramanyan, N. Tsiskaridze, W. Li, A. Gascón, W. Y. Tan, A. Tiwari, N. Shankar, S. A. Seshia, and S. Malik. Reverse engineering digital circuits using structural and functional analyses. *IEEE Transactions on Emerging Topics in Computing*, 2(1):63–80, March 2014.

[135] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik. Reverse engineering digital circuits using functional analysis. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1277–1280, March 2013.

[136] Takeshi Sugawara, Daisuke Suzuki, Ryoichi Fujii, Shigeaki Tawa, Ryohei Hori, Mitsuru Shiozaki, and Takeshi Fujino. Reversing stealthy dopant-level circuits. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 112–126. Springer, 2014.

[137] Takeshi Sugawara, Daisuke Suzuki, Minoru Saeki, Mitsuru Shiozaki, and Takeshi Fujino. On Measurable Side-Channel Leaks Inside ASIC Design Primitives. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 159–178. Springer Berlin Heidelberg, 2013.

[138] G Edward Suh, Dwaine Clarke, Blaise Gassend, Marten Van Dijk, and Srinivas Devadas. AEGIS: Architecture for Tamper-Eevident and Tamper-Resistant Processing. In *International Conference on Supercomputing*, pages 160–171, 2003.

[139] Sun Microsystems, Inc. OpenSPARC Overview. [Online]. Available: `http://www.oracle.com/technetwork/systems/opensparc/index.html`.

[140] Pawel Swierczynski, Marc Fyrbiak, Philipp Koppe, and Christof Paar. FPGA trojans through detecting and weakening of cryptographic primitives. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(8):1236–1249, 2015.

[141] T. Arons et. al. Formal Verification of Backward Compatibility of Microcode. In *CAV*, pages 185–198, 2005.

[142] Ming Tang, Zhenlong Qiu, Weijie Li, Shubo Liu, and Huanguo Zhang. Power Analysis Based Reverse Engineering on the Secret Round Function of Block Ciphers. In *Data and Knowledge Engineering*, pages 175–188. Springer-Verlag, 2012.

[143] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Design Test of Computers*, 27(1):10–25, Jan 2010.

[144] Li Tian. Simple, novel and low cost numerical aperture increasing lens system for high resolution infrared image in backside failure analysis . In *Physical and Failure Analysis of Integrated Circuits (IPFA), 2014 IEEE 21st International Symposium on the*. 2014.

[145] Randy Torrance and Dick James. *The State-of-the-Art in IC Reverse Engineering*, pages 363–381. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[146] Arrigo Triulzi. Pneumonia, shardan, antibiotics and nasty mov: a dead hand's tale. [Online]. Available: `https://www.troopers.de/events/troopers15/449_pneumonia_shardan_antibiotics_and_nasty_mov_a_dead_hands_tale/`, 2015.

[147] Arrigo Triulzi. The chimaera processor. [Online]. Available: `https://www.troopers.de/events/troopers16/655_the_chimaera_processor/`, 2016.

[148] K. Ura, H. Fujioka, and T. Hosokawa. Stroboscopic scanning electron microscope to observe two-dimensional and dynamic potential distribution of semiconductor devices. In *Electron Devices Meeting, 1977 International*, volume 23, pages 502–505, 1977.

[149] Katsumi URA, Hiromu FUJIOKA, and Teruo HOSOKAWA. Picosecond Pulse Stroboscopic Scanning Electron Microscope. *Journal of Electron Microscopy*, 27(4):247–252, 1978.

[150] D. Velenis, M. Stucchi, E. J. Marinissen, B. Swinnen, and E. Beyne. Impact of 3d design choices on manufacturing cost. In *2009 IEEE International Conference on 3D System Integration*, pages 1–5, Sept 2009.

[151] Dennis Vermoen, Marc Witteman, and Georgi N. Gaydadjiev. Reverse engineering Java Card applets using power analysis. In *Proceedings of the 1st IFIP TC6 /WG8.8 /WG11.2 international conference on Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, WISTP'07, pages 138–149. Springer-Verlag, 2007.

[152] B. Victor and K. Keutzer. Bus encoding to prevent crosstalk delay. In *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No.01CH37281)*, pages 57–63, Nov 2001.

[153] Adam Waksman and Simha Sethumadhavan. Silencing hardware backdoors. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, pages 49–63, Washington, DC, USA, 2011. IEEE Computer Society.

[154] Adam Waksman, Matthew Suozzo, and Simha Sethumadhavan. Fanci: Identification of stealthy malicious logic using boolean functional analysis. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 697–708, New York, NY, USA, 2013. ACM.

[155] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*, pages 680–681. Addison-Wesley Publishing Company, USA, 4th edition, 2010.

[156] Maurice V Wilkes. The Best Way to Design an Automatic Calculating Machine. In *The Early British Computer Conferences*, pages 182–184. MIT Press, 1989.

[157] Alexander Wolfe. For Intel, it's a case of FPU all over again. EETimes [Online]. Available: `http://www.fool.com/EETimes/1997/EETimes970516d.htm`, 1997.

[158] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester. A2: Analog malicious hardware. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 18–37, May 2016.

[159] J. Zhang, Feng Yuan, Lingxiao Wei, Zelong Sun, and Q. Xu. Veritrust: Verification for hardware trust. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–8, May 2013.

[160] Jie Zhang, Feng Yuan, and Qiang Xu. Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 153–166, New York, NY, USA, 2014. ACM.

List of Abbreviations

AES Advanced Encryption Standard

AFM Atomic Force Microscopy

ASIC Application Specific Integrated Circuit

AVC Active Voltage Contrast

BEOL Back End of Line

BIB Broad Ion Beam

BGA Ball Grid Array

BIOS Basic Input/Output System

CCVC Capacitive Coupled Voltage Contrast

CISC Complex Instruction Set Computer

CMOS Complementary Metal Oxide Semiconductor

CMP Chemical Mechanical Polishing

CNC Computerized Numerical Control

CPU Central Processing Unit

DFF D-Flipflop

DRAM Dynamic random-access memory

DRC Design Rule Check

DUT Device Under Test

DVC Dynamic Voltage Contrast

EBP E-beam probing

EDA Electronic Design Automation

EEPROM Electrically Erasable Programmable Read-only Memory

EM electro-magnetic

FA Failure Analysis

FEOL Front End of Line

FF Flip Flop

FI Fault Injection

FIB Focused Ion Beam

FMI Fluorescent Microthermal Imaging

FPGA Field Programmable Gate Array

FSM Finite State Machine

GPU Graphics Processing Unit

IC Integrated Circuit

IDU Instruction Decode Unit

I/O Input/Output

IP Intellectual Property

IRDS International Roadmap For Devices And Systems

ISA Instruction Set Architecture

KOH Potassium Hydroxide

LC Liquid Crystal

LVP Laser Voltage Probing

LVS Layout Versus Schematic

MSR Model-specific register

NAIL Numerical Aperture Increasing Lens

OS Operating System

PCB Printed Circuit Board

PC Personal Computer

PEX Parasitic Exctraction

PICA Picosecond Imaging Circuit Analysis

POC Proof of Concept

PVC Passive Voltage Contrast

QE Quantum Efficiency

RAM Random Access Memory

RISC Reduced Instruction Set Computer

ROI Region of Interest

ROM Read-Only Memory

RST Remaining Silicon Thickness

RTL Register Transfer Level

SC Side Channel

SCA Side-Channel Analysis

SCARE Side Channel Analysis for Reverse Engineering

SEM Scanning Electron Microscope

SNR Signal-to-Noise Ratio

SOD spin-on dielectric

SOI Silicon-on-Insulator

SRAM Static Random Access Memory

SSCA Simple Side Channel Analysis

STI Shallow Trench Isolation

TLD Through the Lens Detector

TMAH TetraMethyl Ammonium Hydroxide

TPM Trusted Platform Module

TPU Tensor Processing Unit

uC Microcontroller

UEFI Unified Extensible Firmware Interface

VC Voltage Contrast

VCSCA Voltage Contrast Side Channel Attack

List of Figures

List of Tables

List of Algorithms

About the Author

Personal Data

Name	Christian Kison
Address	Heppenheimerstr 6
	65203 Wiesbaden, Germany
E-Mail	Christian.Kison@rub.de
Date of Birth	June 2nd, 1988
Place of Birth	Soltau, Germany

CV

Since 2014	Research in IT-Forensics and examinations, Federal Criminal Police Office
2014–2018	PhD studies, Chair for Embedded Security, Ruhr-University Bochum
10/2011–02/2014	Student in Informations-Systemtechnik, Technische Universität Braunschweig Master Thesis: *Reverse Engineering of unknown Hardware Structures with SCARE Attacks* Final Grade: 1.1
10/2008 –09/2011	Student in Informations-Systemtechnik, Technische Universität Braunschweig Bachelor Thesis: *Implementierung eines farbbasierten Algorithmus zur Detektion und Klassifizierung von Markern in VHDL für eine schwach programmierbare Architektur* Final Grade: 1.6
2008	Abitur, Gymnasium Soltau

Publications

Peer-Reviewed Conferences, Journals and Workshops

2017 Fyrbiak, Strauß, **Kison**, Wallat, Elson, Rummel, Paar: Hardware Reverse Engineering: Overview and Challenges
IVSW 2017 - IEEE 2nd International Verification and Security Workshop

2017 Koppe, Kollenda, **Kison**, Gawlik, Paar, Holz: Reverse Engineering x86 Processor Microcode
USENIX - 27th USENIX Security Symposium

2015 **Kison**, Frinken, Paar: Finding the AES Bits in the Haystack: Reverse Engineering and SCA Using Voltage Contrast
CHES 2015 - Workshop on Cryptographic Hardware and Embedded Systems

2018 Kollenda, Koppe, Fyrbiak, **Kison**, Paar, Holz: An Exploratory Analysis of Microcode as a Building Block for System Defenses
CCS '18 - Conference on Computer and Communications Security

2019 **Kison**, Awad, Fyrbiak, Paar: Security Implications of Intentional Capacitive Crosstalk
IEEE Transactions on Information Forensics and Security (Journal)

Other Publications

2015 Alendal, **Kison**, modg: got HW crypto? On the (in)security of a Self-Encrypting Drive series
Hardwear.io 2015 - Hardware Security Conference and Training

Book Chapters

2017 Becker, Fyrbiak, **Kison**: Hardware Obfuscation: Techniques and Open Challenges
in "Foundations of Hardware IP Protection", Springer Verlag
ISBN: 978-3-319-50378-3

Conferences and Workshops

Participation in Selected Conferences & Workshops

2015	**CHES'15**	(Saint-Malo, France)
2015	**HardwearIO**	(Eindhoven, Netherlands)
2016	**ISTFA'16**	(Fort Worth, USA)